PERSONAL COMPUTER WORLD

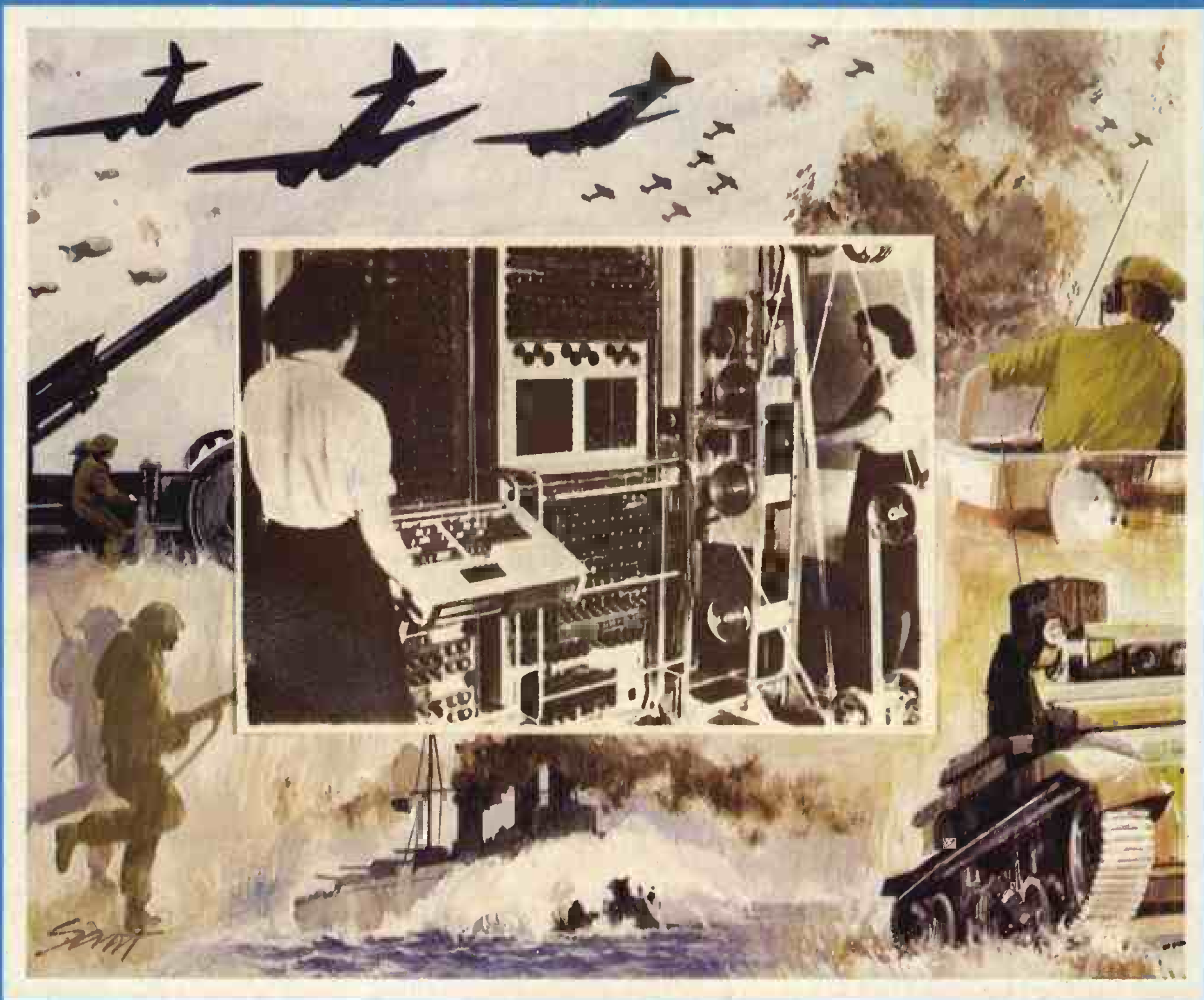
50p U.S.$2.00

DECEMBER 1978

GUIDELINES FOR THE SMALL BUSINESS USER

Europe's first magazine for personal computers for home and business use

Computers and Art
3-D Noughts and Crosses
Brain v Brawn
Fast Cassette Interface



COLOSSUS
Centre of World War II Decoding Web

PERSONAL COMPUTER WORLD

Europe's first magazine for personal computers for home and business use

UK 50p
US $2.00

Vol 1, No 8
December 1978

ISSN: 0142 0232

CONTENTS

The continuation of **Getting it Together** by Mike Banahan, and **Buzzwords** by Peter Reynolds, will appear in the next issue.

Editorial and Advertising Office:
62A Westbourne Grove, London W2
Phone: 01 - 229 5599 (Publisher)
01 - 727 8758 (Editorial and Advertising)

Publisher:
A Zgorelec
Editor:
Meyer N. Solomon
Editorial Assistant:
Roger C. Wilkins
Policy Advisor:
Peter Crofton-Sleigh, FRAS

Consultants:
John Coll, Mike Dennis, Neil Harrison, Charles Sweeten, Patrick Sutton, Michael James, R. W. Davy, David Hebditch, Sheridan Williams, Stephen Castell
Art: Sauveur Laurent Sant, Kathryn Hamme
Secretarial:
Vanessa Blackburn Kiddle

Layout Consultants:
D. Norris, T. Gabos
PCW Photography: Yoshi Imamura, Peter McGee
Typesetting & Artwork: Gilfillan
Cover: Photo montage/painting by Sant. Photo courtesy Public Records Office; Crown copyright.

CONTRIBUTORS:

We welcome interesting articles written simply and clearly. You need not be a specialist to write for us. MS should not be more than 3000 words long, lines double spaced, with wide margins. Line drawings and photographs wherever possible. Enclose a stamped self-addressed envelope if you would like your article returned.

Manufacturers, suppliers and dealers are welcome to contribute technical articles, and send product information, but we are pledged to an independent viewpoint and will publish evaluations and reasoned criticism or praise, space permitting. Naturally there will be right of reply. Views expressed in articles are not necessarily those of Personal Computer World.

We may make arrangements to offer our readers products at special prices, for a limited period, in line with the policy outlined above.

Published monthly by Intra Press, 62A Westbourne Grove, London W2. Phone: 01-229 5599. Contents fully protected by copyright. All rights reserved. Subscription rates: Britain £8 for 12 issues. Prices include postage. USA — $10 for six issues, $20 for 12 issues. Continent and elsewhere: £9.80 for twelve issues. Prices include postage.
Printed by Carlisle Web Offset, 55 Conduit Street, Newtown Trading Estate, Carlisle CA2 7NR. Sole UK Distributors Seymour Press Ltd., 334 Brixton Road, London S.W.9., England. Distribution to specialist shops by Intra Press.

I BYTE

BUT COMPUTERS DON'T

Come and see for yourself at

the BYTE SHOP Ltd

Stockists of the largest range of micro computers in the U.K.

Take the opportunity to experiment with and get to know any of the vast range of micro computers always in stock at The Byte Shop.

Whether you want a micro computer for your home, your business, for industry, for education - or if you'd just like to find out which model you get on with best - you'll find a visit to The Byte Shop a new and invaluable experience.

Call in at The Byte Shop any time from Monday to Saturday. It's right by Gants Hill tube station.

The Byte Shop 426/428 Cranbrook Rd., Gants Hill, Ilford, Essex. Telex 897311
Telephone 01-554 2177

THE RESEARCH MACHINES 380Z COMPUTER SYSTEM

THE RESEARCH MACHINES 380Z
A UNIQUE TOOL FOR RESEARCH AND EDUCATION

Microcomputers are extremely good value. The outright purchase price of a 380Z installation with dual mini floppy disk drives, digital I/O and a real-time clock, is about the same as the annual maintenance cost of a typical laboratory minicomputer. It is worth thinking about!

The **RESEARCH MACHINES 380Z** is an excellent microcomputer for on-line data logging and control. In university departments in general, it is also a very attractive alternative to a central mainframe. Having your own 380Z means an end to fighting the central operating system, immediate feedback of program bugs, no more queuing and a virtually unlimited computing budget. You can program in interactive BASIC or, using our unique Text Editor, run very large programs with a 380Z FORTRAN Compiler. If you already have a minicomputer, you can use your 380Z with a floppy disk system for data capture.

What about Schools and Colleges? You can purchase a 380Z for your Computer Science or Computer Studies department at about the same cost as a terminal. A 380Z has a performance equal to many minicomputers and is ideal for teaching BASIC and Cesil. For A Level machine language instruction, the 380Z has the best software front panel of any computer. This enables a teacher to single-step through programs and observe the effects on registers and memory, using a single keystroke.

WHAT OTHER FEATURES SET THE 380Z APART?

The 380Z with its professional keyboard is a robust, hardwearing piece of equipment that will endure continual handling for years. It has an integral VDU interface – you only have to plug a black and white television into the system in order to provide a display unit – you do not need to buy a separate terminal. The integral VDU interface gives you upper and *lower* case characters and low resolution graphics. Text and graphics can be mixed *anywhere* on the screen. The 380Z has an integral cassette interface, software and hardware, which uses *named* cassette files for both program and data storage. This means that it is easy to store more than one program per cassette.

Owners of a 380Z microcomputer can upgrade their system to include floppy (standard or mini) disk storage and take full advantage of a unique occurence in the history of computing – the CP/M™* industry standard disk operating system. The 380Z uses an 8080 family microprocessor – the Z80 – and this has enabled us to use CP/M. This means that the 380Z user has access to a growing body of CP/M based software, supplied from many independent sources.

380Z mini floppy disk systems are available with the drives mounted in the computer case itself, presenting a compact and tidy installation. The FDS-2 standard floppy disk system uses double-sided disk drives, providing 1 Megabyte of on-line storage.

**Trademark, Digital Research.*

Versions of BASIC are available with the 380Z which automatically provide controlled cassette data files, allow programs to be loaded from paper tape, mark sense card readers or from a mainframe. A disk BASIC is also available with serial and random access to disk files. Most BASICs are available in erasable ROM which will allow for periodic updating.

If you already have a teletype, the 380Z can use this for hard copy or for paper tape input. Alternatively, you can purchase a low cost 380Z compatible printer for under £300, or choose from a range of higher performance printers.

380Z/32K complete with SINGLE MINI FLOPPY DISK SYSTEM MDS-1

£1787.00

380Z/16K System with Keyboard

£965.00

RESEARCH MACHINES Computer Systems are distributed through SINTEL, P.O. Box 75, Chapel Street, Oxford. Telephone: OXFORD (0865) 49791. Please contact SINTEL for the 380Z Information Leaflet. Prices do not include VAT @ 8% or Carriage.


HAPPY XMAS

Editorial

Of machines and men.

Omni is a new American magazine, beautifully produced, with an impressive list of contributing editors. Its first issue is a mixture of science fact, science speculation and science fiction. It's a magazine I'll keep buying — and reading.

There was one article, however, I took exception to. Its title: "Zen; Technology and the split brain". It starts unexceptionably: the hemispheres of the brain are virtually independent of each other, the left dealing with language, the right with intuition, concepts and overviews. The left is "analytical" the right "holistic". Then, in a great leap sideways, we are told the language (left) part of the brain functions sequentially, like a computer. Later, we are told that we are becoming prisoners of the left side, taking the computer as the paradigm for thought. The author coins a new word for this — "computerthink".

We are urged to surrender ourselves to the author's brand of intuition — Zen. To buttress his view that "nonverbal" thought is in decline in (at least) technology, he quotes a professor at the University of Delaware. The professor bemoans the inability, except in architectural faculties, of students to make "isometric views of industrial processes"; and goes on to say, that designers will be "unaware that their nonverbal imagination and sense of fitness have been atrophied by an intellectually impoverished engineering approach." Now, count the number of well worn words of three syllables or more in this one phrase. Swallow in one gulp, if you can, this one dimensional string of words uttered by a champion of "intuition", "creativity", "holism" — no isometric utterance here.

Language is sequential, we are told, devoid of ambient creative meaning. So presumably words like relativity or love are simply letters in a particular sequence preceded and followed by start and stop spaces. No, we are to grasp the wholeness of things by a state of intuition.

Long, long before computers appeared people were thinking in rigid and linear ways. The fact that Galileo and Einstein made intuitive breakaways is no argument against language and mathematics as the prevalent modes of thought; at the very least, intuition springs out of language and logic. Man is not a creature of evolution for nothing. That the nonverbal processes of the mind are largely submerged is no accident. This is the way that the "pressure" is eased occasionally, releasing a jet of intuition, a flame of genius. Otherwise, if we allow "intuition", the "right side", to take over entirely we get blandness, homogeneity, stagnancy. To put it bluntly, addled brains.

At PCW we welcome with open arms the democratisation of the computer, the spread of these logical marvels, tools in the sense that the first stone knife was a tool — no argument there that Man has been using his brain ever since only as a "cutting" instrument.

Creative intuition is a miraculous and beautiful phenomenon. The small computer gives each of us his own laboratory or test-bed to apply that intuition. If we are to be careful of any particular mode of thought it is not of "computerthink" but "glossthink" — a shiny and superficial approach to handling the world. "Glossthink" is a word I have counter-coined and swear never to use again; or I will become yet another victim of a deadly germ — the irrelevant neologism.

Publisher's Letter

Dear Reader,

It is not too early to invite ideas and suggestions for next year's Show. There are questions, for instance, of venue and timing, what features you would like, whether apart from the Chess Tournament there could be competitions like the "mouse in a maze" one being held in the United States. Should we make an effort to get people to exhibit robots? Should we offer a prize to encourage their development in time for the Show? Should we have an Amateur of the Year Award? Could a manufacturer or a private reader build a scale model of a house where the microprocessor is *seen* to be controlling heating, the fire and burglar alarms, the doors and windows, the cooker, the radio and TV set? Apart from the Conferences, should we have three day teach-ins?

No, it is not too early. So I urge everyone, including those companies who want to book stands at next year's show, to write in now.

Letters

ONE MORE FOR THE CASBA

Congratulations on the first few issues of PCW. Much to the distress of my family, I have spent most of my free time totally absorbed in these super issues — I look forward to many more.

I am particularly interested in the idea of a Computer Association for Small Business Applications, and welcome the chance to communicate with other interested persons.

Tyrone Smalls
Director, Zymela Ltd.
Computer Systems Consultancy
163 Lightfoot Road, London, N8

BE SPECIFIC

I would suggest to a would-be purchaser of equipment that as there are so many machines of similar specifications available today, a simple test (such as we use before specifying equipment) be carried out. This involves sending a few technical questions to each of the selected suppliers, all relevant to the system you propose to use, and seeing how long it takes to get a reply. If they can't bother to reply when an order is in view what chance have you got after you buy and pay for the equipment!

P. W. Hutton
Mechanical & Electrical Consulting Engineers
49 Otley Road,
Shipley, West Yorks.

PCW *See David Hebditch's hints in this issue.* PCW.

BRAIN GAIN

First of all — congratulations on an excellent breakthrough, a much needed publication on this side of the water. Keep the software going, it's great to be able to pick other people's brains on common problems for free! I am a programmer by profession, into such things as distributed processing (dedicated micros really) so it's nice to be able to get back to the less tedious side of programming and do it *all* your *own* way. My own system is Z80 based and when I have any software I think will be of merit I will be glad to let you have a copy.

Once again, congratulations.

T. E. Bottomley,
232 Birmingham Road,
Stratford upon Avon, CV37 0AX

ENQUIRY

I am interested in games written in Algol 60 or Fortran: chess, Mastermind, Startrek . . . I would also like to gather information on the applications of computers in banks in totalling accounts, checking balances, reminders etc.

Francis Adegbile,
Tawney 11, Room 14,
University of Essex,
Colchester CO4 3TZ

A SHORTER SUBMARINE

In the article "Submarine Chase", (PCW Vol 1, No. 6), your contributor has set variable S to 1, 2 or 3, and wishes to set variable E to 3, 2 or 1 respectively. To do this he takes 6 lines of code (180 - 230).

I know Tandy Level 1 is pretty basic (sorry!), but surely it can cope with: 180 E=4—S.

C. R. Culpin,
56 Forest Drive West,
Leytonstone,
London, E.11

PCW No claims were made for efficiency of coding. Improvements are *always* welcome; they can only increase the entertainment value of the program PCW.

SHOWDOWN

October PCW. Quote, Chris Howland's "Personal Experience". . . 'never before did I realise that just one silly little glass of wine can blow the works. If you want to give up drinking, start computing.' Unquote. Fine. But I couldn't help noticing that virtually all the exhibitors at the PCW Show, and more than the usual proportion of the visitors (mainly those who at least gave the impression of understanding exhibitors!) were *chain-smoking.* Does computing then merely drive one from drink to tobacco? Shall we see microcomputers sold with an H.M. Government Health Department's Warning: 'Computing can Damage your Health'. Perhaps I had better stick to a non-electronic highly *un*predictable 'pet'.

Main reason I paid my £1 to come to the Show was to have a good look at the computer-controlled model railway by CAP Microsoft. Where was it? Does the Trade Description Act not apply to exhibitions? The only satisfaction I got was from your only Scandinavian exhibitor, Bo Hellstrom, of BhiAB Electronics. He told me of a club in Gotland that had a couple of his processors tied up to their model railway and promised to put me in touch.

Score — Britain 0 Sweden 1 . . . . .

Paul R. King,
25 Fir Tree Way,
Keymer,
Hassocks,
Sussex BN6 8BU

PCW The railway ran on crossed lines! It was not ready on time. There was a notice to that effect at the Show. PCW.

SCHOOLBOY'S SHOWPIECE

I am very interested in computing and am taking A levels in Maths, Physics and Computer Science.

At school we have our own computing club which meets every lunch time and after school. Several of us went to the PCW Show. I found it very interesting as there were a lot of computers on display to play games with. While at the show I tried your "New art form" program (pg 54 PCW Oct.). There is, by the way, a close bracket missing in the listing.

I have written a program using the same idea and tested it at the Show. It is:

```
10 FOR I= —1 TO 1 STEP 2 20 A=INT (RND(2)*8+2)*I
30 IF I= —1 THEN P=999 40 IF I= 1 THEN P=0
50 C=INT(RND(3)*255) 60 P=P+A 70 IF P>999 THEN 100
80 IF P<O THEN 100 90 POKE 32768+P,C:GO TO 60
100 NEXT I:GO TO 10.
```

I very much enjoy reading your magazine and am always looking forward to the next issue.

C. Bulcock,
6 Hillcrest Road,
London, W.3

ADEPT ADAPTOR

I read with interest the two articles on the TRS-80 computer (PCW Nos. 4, 5). You mentioned the problem of having to remove the remote lead from the cassette deck to wind on or rewind. This can be overcome by using an adaptor which plugs in between the remote head and the deck. By simply operating the pushbutton you can wind or rewind at will. This does not invalidate the guarantee as would fitting a switch on the deck. The adaptor is available from: Avon Technical Services, 49 Mearns, High Littleton, Nr. Bristol, for £2.29 inclusive of p. & p.

P. R. Cannard,
1 Mead Way,
Bristol

DISPELLING A MYTH

Your potted history of computing (Vol. 1, No. 1), perpetuates a popular myth that electronic computing originated in the U.S.A. The *first* commercially available memory unit was developed by Prof. F. C. Williams of Manchester University, and the IBM 701 and 702, to which you refer, built in 1953, incorporated Williams tubes built under license.

The Mark 1 computer, built at Manchester University, was the *first* stored program computer and the *first* computer program written was written by Dr. Kilburn.

The LEO conceived by Standingford and Thompson of Lyons Caterers was the *first* computer designed and built for commercial work.

The *first* transistorised computer was built at Manchester University and the production version, the MV950, was made by Metropolitan Vickers.

The Atlas computer, built about 1959, had a computing power equivalent to four IBM 7094s and at that time was the most powerful computer in the world, a lead it held for ten years.

Even the printed circuit was conceived by an Englishman in 1936.

G. A. R. Taylor,
Lecturer,
Department of Chemistry, Computing and Applied Sciences,
Warley College of Technology,
Smethwick.

THE LITTLEST COMPUTERS?

I read Derek Chown's article in the September edition of PCW with interest (page 24), but I was surprised to see the statement 'While programmable calculators are definitely not computers they do facilitate real programming with conditional branching, loops and subroutines'.

I run the British *TI Users' Club* which exists for the exchange of programs and information for the whole range of Texas Instruments programmable calculators. We are neither sponsored nor supported by Texas Instruments Limited.

What is a machine which is capable of conditional branching, loops and subroutines if it is *not* a computer?

The only real differences between small pocket programmables and small 'conventional' computers are the restricted program and data storage available and the low operating speed.

They are definitely easier to program than microprocessors using machine or assembly languages and, for scientific uses, they have many pre-programmed functions : trigs, logs etc. On many small BASIC interpreters these functions are provided but may be of low accuracy — 1 part in 10^7; my TI59 is generally accurate to about 1 part in $10^{10}/^{11}$.

I would assert that programmable calculators are capable of almost all normal computer functions and they have two enormous advantages — they are portable and cheap (ca £200 maximum).

P.S. Information on the BTIUC is available from this address on receipt of an s.a.e.

P. R. Rowley,
2 Woodside Crescent,
Clayton,
Newcastle-under-Lyme,
Staffs ST5 4BW

THE READER AS CRITIC

The following comments are provoked by your October issue (Volume 1, No. 6):—

1. When recording data it should be formatted into blocks and each block concluded with Cyclic Redundancy Check (C.R.C.) to enable better than 99.998% detection of all errors. When reading such data, it should be *quarantined* in a buffer and not released for use until validated by the C.R.C. My own monitor incorporates a Tape Operating System that automatically provides the above features. Most systems do not.

The MINMON Z80 monitor appears to suffer common limitations, including:—

a) Object Code Blocks protected by a check sum that provides only 99.6% detection of errors; b) Use of the Object Code as received BEFORE checksum validation.

Due to 'b' 85% of all errors will be detected *after* the wrong value of object code has been stored in memory, and upon detection of the error it is sufficient to re-read and re-store; *but* 15% of all errors will be in the address or byte count, and will be detected *after* storing the "right" values in the *wrong places*, thereby over-writing and destroying what used to be there. You may have just over-written and destroyed a piece of programme that was loaded into memory half an hour ago. I therefore disagree with Neil Harrison, and suggest that when a check-sum error is detected the whole programme must be re-loaded, not just the last data block, *and* all previously loaded tapes must also be re-loaded as well if you want to be sure.

2. I am horrified by the "DECODE" routine by Stephen Collins. If the last command in the table is RUBBISH and the character string on the stack is RUBBISH followed by 1 unused byte followed by the subroutine return address, it will work — just; *but* if the character string on the stack is RUB BARB the match terminates with accumulator 'A' holding 'A' and accumulator 'B' holding 'I'. The DCODE3 loop then winds the index down until it is pointing at 'H', with the stack pointer still above 'R' — we then hit DCODE5 and the RTS instructions causes 'RB' to be pulled from the stack and put into the programme counter. Instead of going back to the correct return address, it goes to address 5242 (the Hexadecimal values corresponding to the ASCII characters RB). If you mis-type the character string the programme runs wild in some unpredictable fashion dependent upon the characters actually typed.

3. Dr. Michael Hendry was concerned about lack of confidentiality if patients' medical records were stored on a central computer.

I suggest there need be no loss of confidentiality if each record is identified by an arbitrary reference number and the name of the Doctor, who (with the aid of his own personal computer perhaps), can determine from the reference number the name and address of the patient. Although Government Officials might mis-use the *data base* to obtain statistics upon un-married mothers to facilitate planning for future "single parent" housing requirements etc., they cannot obtain the *identity* of any one girl.

The medical record may include year of birth. Exact date of birth is best omitted if you wish to prevent effective use of Somerset House records to identify (with name and address) the subject of a medical record.

By writing a prescription with the patient's name and address, the Doctor is in breach of confidence every time a N.H.S. filing clerk reads the prescriptions. This may be the reason for the traditional lack of legibility in the writing! It would become necessary for the doctor to put the appropriate reference number on the prescription, instead of the patient's name and address.

With such a system, it is possible for a central computer to detect *and avert* future Thalidomide type situations more rapidly than the last time, with far fewer casualties.

A. J. Borer, C.Eng., M.I.E.E., B.Sc.
22 Deerfold,
Woodfield Park,
Astley Park,
Chorley,
Lancashire

PCW We appreciate Mr. Borer's criticism. It is of the best kind, helpful and informative. One should not forget that Stephen Collins presented his article not as an "expert" but as someone in the learning process. PCW

A PLEASANT QUIBBLE

I am a mini/micro assembly language programmer working in real-time applications. I am generally very pleased with the first few issues of PCW. My only quibble concerns the printing and other errors that slip through; a technical magazine *must* be accurate, particularly if people are to use the solutions suggested. To this end, I would like to point out some errors in Mike Lord's otherwise excellent articles on the PDP—11s. (PCW Nos 3 & 5).

1) The instruction CLR 1234 does clear location 1234 as stated, but the instruction uses PC relative (indexed by register 7) addressing for position independency. To use absolute addressing (mode 3), write CLR@#1234.

2) The DEC standard names for registers 6 and 7 are SP (stack pointer) and PC (program counter) respectively.

3) Instructions of the type MOV PC,TEMP are flagged as errors by the PDP—11 assemblers, on the grounds that different PDP—11 processors give different results. The correct method is to go via a register:

```
MOV     PC,R5
MOV     R5,TEMP
```

The value stored is that of the PC at the start of the second MOV instructions.

4) There are four bugs in Mike Lord's example program, as follows:

a) The strings are called STRINGA and STRINGB. PDP—11 assemblers only look at the first six letters of a symbol, so these are treated as the same address.

b) In the instruction CMP R1,#STRINGA+1024 the number will normally be treated as Octal 1234 and not decimal. To ensure decimal assembly, use a 'decimal point' thus: CMP R1,#STRINGA+1024. ; Comments.

c) In the instruction shown in (b), surely we should say CMP R1,#STRINGB+1024.

d) The instruction BLT LOOP is only valid if the strings are guaranteed to be stored below address 100000 (octal). The compare instruction does a subtraction, discarding the result after setting the condition codes; it is possible that overflow may occur. The branch 'BLT' tests for this condition as it is a two's complement test. Thus 100000 (minus 32768.) is LESS than 1000 (plus 512.), but is clearly not a lower address. For correct results, use the unsigned branch:

```
BLO   LOOP    ; Branch if LOwer to LOOP
```

P.S. Long live the PDP—11!

Rupert Steele,
17 Lawrie Park Crescent,
London S.E.26

PCW Thank you. We have the best readers in the world. PCW

THE EUROPA BUS

As a professional microcomputer engineer I feel compelled to reply to Mr. Stiles's letter in the August issue of PCW.

I have in my work encountered a number of microcomputer bus systems used by various manufacturers, and so claim hard experience. I have learnt that 'dedicated' slots on a 'bus' are more trouble than they are worth. I have learnt that manufacturers claiming to 'second scource' boards for a bus are not always reliable. I have learnt that the simpler a bus is the better chance it has of behaving as it was designed to.

However, let me clear up a number of points Mr. Stiles raised in his letter, before going on. On page 26 of the 1977 (please note the year) edition of the supplement to the Texas TTL data book there is a brief description of four octal bus tranceivers. These are available in both inverting and non inverting types, with tri-state *and* open collector outputs available for each. In the TTL data book itself there are such devices as the 74LS240 which is an inverting tri-state bus driver. The 74LS240 through to the LS245 form one family of tri-state bus drivers, the Fairchild 74LS540s another and the Texas 74LS640s a third. Similar devices are manufactured by National Semiconductor but do not have type numbers that are so memorable. I do not think that it would be unreasonable to say that any device the amateur is likely to need for building a personal computer is probably available, he just has to look deep enough. The E78 has not been designed for the Z80 with the 6800 as a second choice. Intel make use of the $\overline{RD}$, $\overline{WR}$ rather than RD/$\overline{WR}$ on a number of their microcomputer support chips. In fact it is possible to show that the use of $\overline{RD}$, $\overline{WR}$, $\overline{MEMRQ}$ and $\overline{IORQ}$ can give an extra 50 nanoseconds in which to decode the type of bus cycle in some cases. The Intel $\overline{MULTIBUS}$ which is popular in America makes use of $\overline{MEMRD}$, $\overline{MEMWR}$, $\overline{IORD}$ and $\overline{IOWR}$ type of signals. No doubt if E78 had used this type of signalling Mr. Stiles would be claiming that E78 was favouring Intel devices!

In his penultimate paragraph Mr Stiles claims that standard modules are available for the S-100 bus. I would like to point out that S-100 is *not* a standard. It is a collection of part standards from different manufacturers. There is no guarantee that an I/O board from one scource will work with a CPU board from another. As an example, IMSAI and Polymorphic both claim to produce S-100 boards; their boards are, however, incompatible. An attempt has been made to define a S-100 standard, but is self contradictory and will, no doubt, render all boards made before it is adopted incompatible with those made after. One of the grave faults of S-100 was to have unused lines that could be used for undefined purposes. Unfortunately everyone chose the same lines for different uses. For this reason I would not like to see *any* lines made available for the user. As to the appearance of standard modules on the market, I have asked Mr. Cottis, one of the designers of this bus, if he would publish the circuit of one of his memory cards. (Everyone needs memory). Hopefully some PCB manufacturer would then produce a board for this and we would have taken the first step towards making E78 a success.

Should anyone be in dire need of a system dependent signal or I/O path, I would recommend him to take such signal from a connector on the front of the card, as is done on the better professional cards. The example of the 9900 CRU line that Mr. Stiles cites is not totally relevant. The 9900 can make use, as can *all* microprocessors, of memory mapped I/O, the CRU lines are just a complicated and difficult to use (certainly for the amateur, since it is for many professionals I know of) method of achieving I/O.

The signals on an E78 bus are sufficient to cope with multiprocessor systems. Since these are liable to be difficult to develop from both the hardware and the software aspects (as may be illustrated by various texts on the principles of multiprocessing) I can understand Mr. Cottis not wanting to baffle the beginner by explaining the protocol E78 uses for concurrent processing. (An excellent introduction would be any one of P. Brinch Hansen's books on the subject — ask your local library to obtain one for you).

Finally I would like to point out that the use of indirect connectors appeals more to the professional than the amateur — the professional can use flow-solder techniques for mass production the amateur can not.

Anthony J. Aylward, B.Sc.
194 Balmoral Road,
Gillingham,
Kent

PCW Mr. Aylward expressed his preference for publishing this letter rather than his letter of criticism and comment (see No. 6) PCW.

ERROR MESSAGE

There is one small error in 'The Hard Keyboard' which may cause some confusion. The output from F/F1 which is used to blank digit 2 and resets the R–S flip-flop is the Q output. This is shown correctly in Figure 1 but incorrectly in Figure 2.

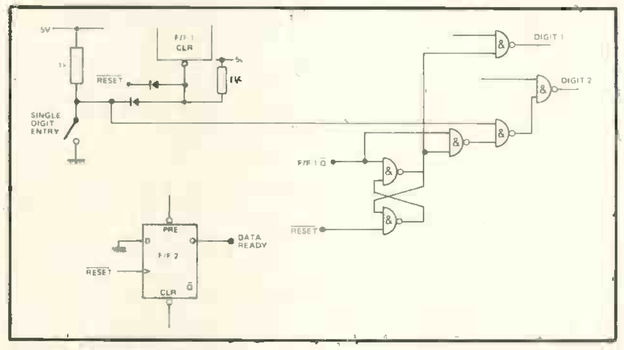
I have made some small changes to the original circuit which allow the reset button to clear the data ready signal and also allow single digit entry which may be required by some programs.

To clear the data ready signal the D input of F/F2 is held at logic 'O' and the clock input is connected to the reset button.

For single digit entry the CLEAR input of F/F1 is held at logic 'O', thus F/F1 Q is always logic 'O', the data is always clocked into latch 2 and the data ready signal is set after every key depression. During single digit entry digit 2 of the display is permanently enabled so that pressing the reset button only blanks digit 1.

W. McIvor,
2 George Street,
Accrington,
Lancs. BB5 0HD

PCW The article referred to appeared in No. 5. PCW



Tid Bits PRODUCTS . . . . COMPANY NEWS . . .

OHIO SCIENTIFIC APPOINTS UK DEALER NETWORK
Dealers have now been appointed for the Ohio Scientific range of microcomputer products for the whole of the UK. Ohio are the world's largest full line microcomputer company – from computers on a board to hard disk (74 Mb) systems.

Superboard II – computer on a board with BASIC in ROM for less than £300.00
OSI's latest product – the 600 board – offers 8K full-feature BASIC in ROM, QWERTY keyboard, 4K RAM, video and Audio interfaces – all on one board for £296 + VAT. Additions needed are: power supply, RF converter, TV and cassette recorder. Expansion capabilities include: further 28 K RAM, port adapter for modem or printer, and dual minifloppy interface.

The Superboard II (600 board) uses the 6502 microprocessor – and includes Microsoft BASIC in ROM which "runs faster than all currently available personal computer BASIC's".

For the North of England and Midlands contact:
U – Microcomputers,
P.O. Box 24,
Northwich,
Cheshire

OHIO SCIENTIFIC Superboard II

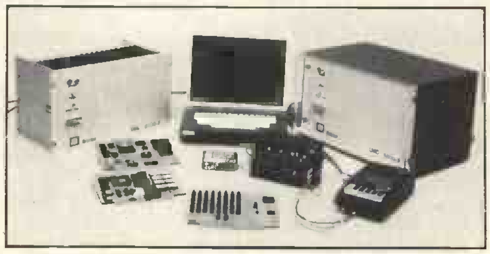
LOW COST MICROCOMPUTER FEATURES ADVANCED DEBUGGER AND CRASH-PROOF BASIC INTERPRETER
Limrose's LMC 6800-2 Microcomputer System, based on the Motorola 6800 microprocessor and compatible with the EXORciser bus, features an advanced new facility for hardware and software debugging. It also features a "crash-proof" BASIC Language Interpreter, which makes it advantageous for use in schools and colleges as well as for industrial control.

The debugging facility, called the "trap", provides a built-in logic analyser at no extra cost and allows one to examine cycle-by-cycle operation of the microcomputer. By trapping up to 250 machine cycles, one can single-step the processor "backwards" in time to establish the exact execution of a faulty programme prior to its crash during development work. This facility is useful for debugging industrial control programmes written either in machine code or assembly code. The system can be supplied fitted with Input/Output ports to meet industrial requirements and with the ROM-based BASIC Interpreter and Assembler.

The basic system consists of just three printed circuit boards, and is ideal for use in schools for computer science studies. The CPU card holds the 8K BASIC Interpreter in read-only memory, the 4K RAM card provides sufficient memory for writing fairly long programmes in BASIC or machine code, and the third card provides the "trap" facility and an interface to the Teletype or Visual Display Unit. A low-cost cassette interface is also available for storing programmes and a floppy disc controller and interface is expected to be ready in a few months' time.

The LMC 6800-2 Microcomputer is available in either kit form, or fully assembled and tested. Price for the minimum system, with 4K RAM, the "trap" facility, 16 lines of digital I/O with interrupt capability, and interface for the VDU or Teleprinter, with all necessary power supplies, motherboard and the 19" rack mounting frame is £290 in kit form. Further information from Limrose at Northwich (STD 0606) 41696.

LIMROSE ELECTRONICS LIMITED,
Microprocessor Division,
241-243 Manchester Road,
Northwich,
Cheshire, England CW9 7NE
Tel. 0606 41696



The Limrose 6800-2

Off-the-shelf bubble-memory terminals from TI
Now available on an off-the-shelf basis from Texas Instruments, the Model 765 portable bubble-memory terminal offers a unique method of data entry, editing and storage for commercial applications such as remote sales-order entry, computer time-sharing systems and newspaper reporting. Because the terminal's bubble memory retains data even when the power is switched off, information from a variety of sources can be stored in the terminal for as long as required, and then transmitted in a single batch over a normal telephone line using the built-in acoustic coupler.

The Model 765's bubble memory has no moving parts and requires no external storage media. In addition, the Model 765 has specifically been designed for ease of use in the normal business environment, with a standard typewriter-like keyboard and simple English-language commands.

Using TI's 'Silent 700'* thermal-printer technique, the Model 765 is a full-capability 30-characters-per-second terminal with a full ASCII keyboard, a powerful command mode, and a file management system.

Further information:
Dave Monk
Texas Instruments Limited,
Digital Systems Division,
Manton Lane,
Bedford MK41 7PA
Tel: (0234) 67466

The Texas Instruments Model 765 portable bubble-memory data terminal, which retains data even when switched off.

Hands-on computer experience for Dulwich College
Dulwich College in South-East London has purchased a Digital Equipment PDP-11/34 computer system which is the largest 'in-school' computer system to be installed in the U.K. to date.

The PDP-11/34 was chosen, running a Resource Sharing/ Time Sharing (RSTS/E) system which offers timesharing facilities for up to 32-users as well as the use of resources such as disks, printer and magnetic tape. The Dulwich system incorpor-

ates eight terminals, two disk drives, a fast printer and a magnetic tape unit.
Further information:
Digital Equipment Co. Limited,
Digital House,
Kings Road,
Reading, Berks.
Tel: (0734) 583555

The PDP 11/34 at Dulwich College

Apple in the Kitchen
It seems to be a fact of life that designing a kitchen takes a long time. The KITCHEN SHOP in Stratford upon Avon has recently installed a system that allows you to design and draw out a kitchen plan in less than 15 minutes. If the housewife changes her mind any unit can be removed, replaced or moved elsewhere and the new layout redrawn in about 20 seconds. Doors, windows and any fixed items (e.g. a boiler) can also be drawn. The system allows the designer to choose the scale he requires for each kitchen. At any time during the design stage a list of units and the cost for any manufactures can be displayed.

This very simple system uses a domestic television, small disk unit and a micro-computer (the Apple) that weighs less than eleven pounds. The cost of the micro-computer, disk drive and the programs to do the above is 3,000 pounds.

The system can be expanded to include a printer that will print the agreement for the customer, print the parts required for the supplier and draw out the kitchen plan for the installer.

Demonstrations by arrangement, telephone Stratford upon Avon (STD) 0789) 66237.

The Apple package is supplied by:
Templeman Software Services,
P.O. Box 7,
Stratford-upon-Avon,
Warwickshire CV37 8LR

Apple Application

Research Machines are now distributing low cost ALGOL–60 compilers for the PDP11, PDP8 and PDP12 computers
All these ALGOL compilers use identical 'Front Ends' which produce a machine independent intermediate code. This means that programmes written for any one machine can be run virtually without modification on any of the others, protecting the customer's software investment.

A CP/M version for Z80 based microcomputers will be released soon. The cost is £250 for either PDP11 or PDP8 versions, and will be £99 for the CP/M / Z80 version. All versions are extremely core efficient and run either stand-alone or under any of the DEC 11 or 8 operating systems. The minimum requirement is 8K words for the PDP versions, 20K bytes for the CP/M /Z80 version.

The ALGOL has been distributed for five years by Dr. Roger Abbott who wrote it. There are now over 100 users world-wide, including British Steel Corporation, the Ministry of Defence and the Technical University of Berlin. It is being implemented on the Z80 at Research Machines by Dr. Colin Morgan who had extensive experience using the ALGOL to process data from Nimbus satellites.
For further information contact:
Michael O'Regan,
Research Machines Ltd.,
209 Cowley Road,
P.O. Box 75,
Oxford.
Tel: 0865 49791

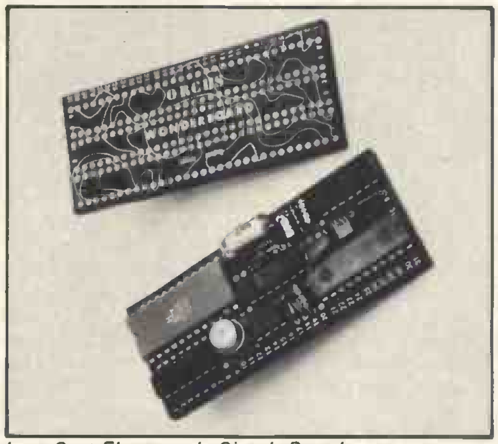
Microdigital, the new Computer Shop at 25 Brunswick Street, Liverpool, announces that it will be pleased to forward copies of its catalogue to enquirers.
Write, or phone:
Bruce Everiss at 051 236 0707

LOW COST ELASTOMERIC CIRCUIT BOARDS
The new solderless circuit boards "WONDERBOARD" allow the user to insert one component lead and up to 6 interconnection wires into each conductive contact. The boards are essentially the same dimensions as a printed circuit board.

These boards can be used for prototypes or for production. They replace Plated Hole PC boards, multilayers, wirewrap as well as breadboards. Reliability is assured by the seal formed between component leads and the conductive elastomeric material when components are inserted.

Any desired board format can be made up by using cyanoacrylate adhesive to join smaller boards together. Two standard models are presently available "Small Wonder" has a capacity of 12 ICs (DIL-14) and "Big Wonder" which has a capacity of 481Cs. Wonderboards accept all sizes of IC packages up to 60 pins as well as discrete components.

Wonderboards are reusable and can be used as circuit boards, as IC sockets or as Hardwire Programmers.
This product is available in the UK exclusively from:
Charcroft Electronics Ltd.,
Charcroft House,
Sturmer,
Haverhill,
Suffolk CB9 7XR
Tel: (0440) 5700



Low Cost Elastomeric Circuit Boards

ANOTHER VIEWDATA FIRST FOR ITT
ITT were the first television manufacturers to test Viewdata from an overseas base, in 1976. The following year they were again first, introducing a tuner unit operable with equal facility whether housed in its TV receiver or taken from the set for remote control of all functions, including Viewdata keying. Last Spring brought another first in the shape of a TV-linked printer producing pictures on paper at the touch of a button, showing Viewdata and Teletext screen displays in the form of immediate information print-outs.

Now comes another first in the same field, a development that stems from an ITT feasibility study now in progress. This is aimed at production of a *hard-copy printer* to sell at a price acceptable to buyers of Viewdata television receivers. With world markets in mind, the printer design must ideally be made to cater for a wide range of foreign language signs and accents, but until now, printers able to reproduce screen images from a television have been limited to the Latin-based alphabet and some associated signs.

In ITT's latest prototype printer, this range has been widened to cover a total not far short of 200 accented characters, thus opening up full on-the-spot print-out facilities in 37 languages native to about 500 million Europeans.

Some of the Viewdata sets made by ITT already obey the Post Office computers for reproducing these newly-stored word components. This breakthrough in printer design was achieved in close collaboration between various of ITT's research and development teams and Post Office engineers.

The extensive work involved in research and development for identifying the many characters covered by the new multiple-character range, and for incorporating them in Viewdata computers, was carried out by Post Office specialists at Martlesham.

For more information:
ITT Consumer Products (UK) Ltd.,
Chester Hall Lane, Basildon, Essex.

Livingston Hire Ltd., has entered the microprocessor field and now offers hiring facilities for two systems: The Motorola Exorciser and the Intel MDS221.

Details from:
Bob Mundy or Peter King,
Shirley House,
27 Camden Road,
London NW1 9NR
Tel: 01 - 267 3262

One of the microprocessor learning systems available on short term rental from Livingston Hire: The MOTOROLA EXORCISER

ADLER TA21 MULTI-PURPOSE DATA CAPTURE SYSTEM. TO BE LAUNCHED AT COMPEC '78

In its basic configuration (the system is cassette-programmed), the TA21 will be available in the UK at £3,300. For an additional £700, the system will incorporate full telecommunication facilities.

Suited for applications in both large and small companies, this extremely versatile computer provides facilities for OCR print-out, off-line data capture and validation, as well as on-line communications, making it unique in both concept and price.

For further details, contact:
ADLER BUSINESS SYSTEMS LTD
Jordan House,
47 Brunswick Place,
London N1
Telephone: 01 - 251 2712

THE ADLER TA21 Low Cost, Multi-Purpose Data Capture System for £3,300

MICROCOMPUTER ARCHITECTURE

AN INTRODUCTION – Patrick Sutton

Microcomputer architecture is, as the name suggests, the structure and inter-relationships of the various functional units forming a microcomputer. Figure 1 shows the basic layout of a microcomputer system.

The central processor unit (C.P.U.) of the system, is the part of the machine which controls the other units in the system and manipulates data within the machine, by following a sequence of instructions known as a programme. In microcomputers, this unit is in the form of a single large scale integrated (LSI) circuit – the microprocessor unit or MPU. An LSI circuit is a complex circuit which is formed on a single chip of silicon by means of a photographic process. For instance, the Intel 8085 MPU has 6,200 transistors, can execute 770,000 instructions a second and is formed on a single chip of silicon 0.164 inch by 0.222 inch.

Information (data and instructions) within the computer, consists of binary digits (bits) which are grouped to form functional pieces of information known as words. Most present-day MPUs work with information in the form of 8-bit groups, although 16-bit MPUs are becoming more readily available. In general an 8-bit group is known as a bite (byte) and 16 bits and above known as words. Although, with the increasing popularity of the 8-bit processor, byte and word are often synonymous. Amusingly enough, 4 bits are sometimes called a nibble.

The memory in a microcomputer consists of storage elements for the words, each word having a uniquely defined location within the memory, known as its address. A microcomputer's memory is usually in the form of integrated circuits and there are numerous types.

Random access memory or RAM is a general store. Random access implies that it takes the MPU the same time to exchange information with the memory, irrespective of the position of that piece of information within the memory. The time taken is known as access time. Random access storage contrasts with serial storage, where access time is dependent upon the position of information within the store.

The use of a dictionary may be considered a form of random access, whereas a cassette tape is a serial store. In order to access information on a cassette tape, you have to wind the tape from its present position to the position of the required information, passing through all the intervening information one piece at a time, whether it is read or not. The advantages of random access, in terms of speed, can be seen if you imagine having to use a dictionary in a serial fashion.

The major disadvantage of RAM is that when the power is turned off, all the information stored is lost; the RAM is said to be volatile. This is where read only memory or ROM has the edge. This is a form of memory which is not generally accessible to the MPU for writing into – it is read only. When the power is turned off, ROM still contains the previously stored information; it is non-volatile. It should be noted that ROM is a random access and not serial when it is read.

In order to store information in ROM, special programming devices are required. This type of memory is used to store data tables or frequently used programmes such as the monitor programmes found in many microcomputers.

Let us now examine the MPU in a little more detail. The MPU contains registers, an arithmetic logic unit (ALU) and control circuitry.

REGISTERS

These are temporary storage elements within the MPU, some of which have dedicated tasks, and some of which are for more general use. The following registers are frequently encountered in MPU structures:–

Accumulator – This is a general purpose register closely inter-connected with the ALU and is the register used for arithmetic, logical and rotate/shift type instructions. The majority of data manipulation is performed using the accumulator.

Address Register – This register is used both to build and manipulate addresses to be used by a particular instruction. It allows the address to be constructed *during* the execution of an instruction. For instance, if an instruction requires data to be loaded from memory, the location of the data in memory, its address, is placed in the address register.

Index Register – This register can be used, in conjunction with the address register, to form an address during the execution of an instruction – a form of addressing known as indexed addressing. The index register can also be used to control programme loops; that is in controlling the number of times a particular sequence of instructions is to be executed. To facilitate such uses, the index register often has associated instructions such as "increment or decrement index register," and "test the contents of the index register".

Programme Counter – This special purpose register stores the address of the next instruction to be fetched, decoded and executed by MPU. This differs from the address register in that it points to the location of the next instruction and not an address to be used *by* the instruction.

Certain instructions manipulate the programme counter directly, allowing segments of the programme to be performed in a sequence defined by the programme itself, or the intermediate results of the processing. These instructions are the jump, branch and jump to sub-routine instructions. These instructions may be unconditional or conditional; that is, execution of the jump or branch may be dependent upon certain conditions being met during the running of the programme. Unless one of these instructions was the last to be executed, normal instruction fetch, with associated incrementing of the programme counter, is performed automatically by the MPU upon completion of the last instruction. If one of the special programme counter instructions is executed, and its condition, if any, is satisfied then the programme counter is loaded with the address specified in the instruction, and further instructions are fetched from the *new* address.

Stack Pointer – This register, which may be altered directly under programme control, contains (points to) the address of an area in memory designated as the stack. This area, which can be anywhere within the computer memory, can be used as a data store or in-directly as a store for (saving) register values during a subroutine jump. Use of stack orientated instructions increases programming efficiency, as the address for data storage or retrieval is implied by the instruction and saved in the stack pointer. Thus single word instructions can store or recall data from anywhere in the memory. The stack orientated store or recall instructions automatically adjust the stack pointer to point to the next relevant memory location, thereby offering even greater programming efficiency.

Instruction Register — An instruction code is loaded into this register after an instruction fetch operation, and prior to decoding in the decoder. This register is not directly available for programming. It is said to be "transparent" to the user.

General Purpose Registers — These are used by the MPU to store intermediate results during instruction execution. In most cases these are transparent to the user, but some MPUs, for example the Z80, do have additional registers available for programme use.

In 8-bit MPUs, the accumulators and instruction registers are usually composed of 8 bits. The other registers are usually 16 bits long. In the case of those registers pertaining to address manipulations, the use of 16 bits allows a larger memory to be dealt with.

ARITHMETIC LOGIC UNIT (ALU)

This is the unit within the MPU, which performs the arithmetic and logic operations on data. This unit usually contains an adder and a subtractor circuit, and more advanced ALUs cater for hardware multiply and divide. Provision is made for Boolean logic operations such as AND, OR etc. Closely associated with the ALU are *flag bits*. These are indicators of such conditions as arithmetic overflow (number generated by operation is too large for the given word length), carry/borrow indicators for addition and subtraction operations, and data type indicators such as negative and zero flags.

A NOTE ON INTERRUPTS

Another flag usually found in the MPU is an interrupt flag. An interrupt is a provision made in an MPU allowing a peripheral device to stop the processor's current task and redirect it to another routine. This routine is called an **interrupt service routine**. After completion of the interrupt service routine, the processor carries on with its previous task. The interrupt flag is set on receipt of an interrupt, and is cleared on return to the current programme, after the interrupt has been serviced. Some interrupts may be *masked;* that is, prevented from stopping the current task, by setting this flag directly under programme control.

The advantage of interrupt facilities is that the MPU may perform another task whilst a slower device, such as a printer is operating. The slower device will signal the MPU, using the interrupt facility, when it requires attention. Careful use of interrupts can greatly increase the computer's efficiency.

CONTROL CIRCUITRY

This unit, by utilising the timing signal or clock input to the MPU, maintains the correct sequencing and timing of operations within the MPU during the execution of an instruction. It also ensures the correct sequence of actions after an interrupt has been received. This circuitry is totally transparent to the user.

The MPU communicates with its own devices and with the outside world through devices known as input/output ports (I/O). In the microcomputer, these devices are varied, depending upon the form the communication takes. Let us now examine some of the commoner I/O ports and communication formats.

PARALLEL INPUT/OUTPUT

The parallel input/output (PIO) controller, sometimes known as a peripheral interface adapter or PIA, is a device designed to interface or link the computer to peripherals which transmit and receive data in a parallel format. In parallel data transmission, the whole data word is transmitted at once and therefore a separate wire is required for each bit of the word. Parallel transmission is usually used when the peripheral is close to the computer, as the cost of individual data-bit wires becomes prohibitive over long distances. Peripherals which can operate in parallel include printers, keyboards, paper tape punches etc.

The general structure of these devices can be seen by examining one such device produced by Zilog — the Z80 PIO. This, as are other parallel I/O devices, is a programmable unit which contains two ports each of 8 bits. Both these ports can be programmed as an input or an output and one port can operate bidirectionally. Control information can be passed to or from a peripheral by

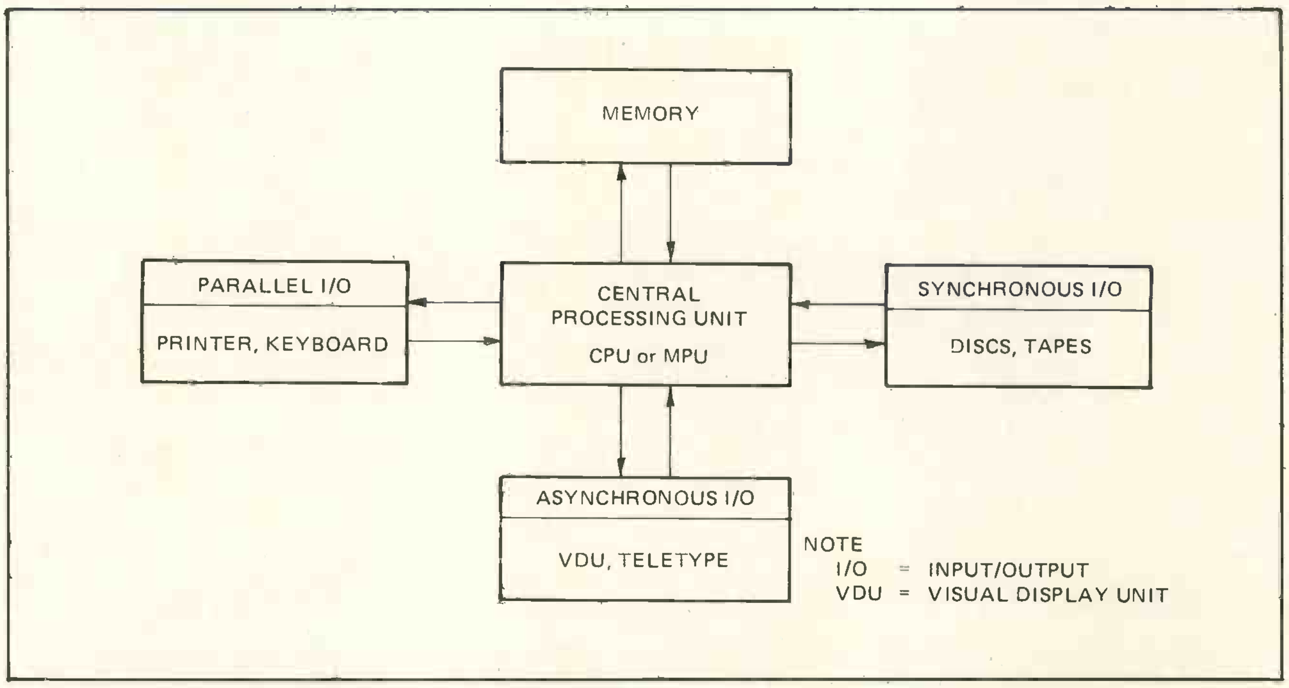


Figure 1. A Simplified Block Diagram of a Microcomputer

using these ports or the interrupt-driven handshake lines. *Handshaking* is a term for communication between a peripheral and the MPU in the form of data ready to transmit and data received signals. In this device, these are interrupt-driven to increase the efficiency of communication, as mentioned earlier.

Another facility available in most of these devices is the ability to programme each of the 8-bit ports in a bit mode; that is each bit may be selected as either an input or an output, in isolation from the other bits in the port.

The great versatility of these programmable devices makes interfacing the computer to a peripheral simple and efficient.

ASYNCHRONOUS SERIAL INPUT/OUTPUT.

An asynchronous communications interface adapter or ACIA, such as that produced by Motorola (MC 6850), is a device used to transmit and receive *serial* data to and from peripheral devices. As most microcomputers are parallel or bus orientated, there is a need to convert this parallel data into serial format; a function performed by serial input/output ports. Serial data is data transmitted one bit at a time along a single line. Serial transmission of data is usually adopted where the peripheral and computer are widely separated and the use of separate lines for each bit would prove too expensive.

Asynchronous serial transmission is characterised by the transmission of functional groups of bits down the line serially. Each functional group of bits, sometimes called a character, is separated from the next group by the inclusion of start and stop bits signalling the beginning and end of the transmitted character. Typical devices employing such modes of transmission are TELETYPE© and visual display units (VDU's). In fact, the asynchronous format is almost always used in serial communications to and from a human operator i.e. via a keyboard.

The Motorola ACIA has the ability to detect or produce the start and stop bits, depending upon whether it is transmitting or receiving the data. Error detection is built into this and similar devices, as well as the facility for programming the number of stop bits to be used. The Motorola chip is capable of transmitting data at up to 500,000 bits/sec.

A universal asynchronous receive and transmit device (UART) is, as the name suggests, a device capable of communicating asynchronous serial data in all common format types, both to and from the computer.

SYNCHRONOUS SERIAL INPUT/OUTPUT

In synchronous serial transmission, entire blocks of characters are transferred down a single line, in a serial fashion, without individual start and stop bits for each transmitted character. The MC6852 chip produced by Motorola, is capable of interfacing such signals to the computer. Synchronous transmission is more efficient than asynchronous transmission, but it does make greater demands upon the exact timing of the individual characters, due to the lack of the start and stop bits.

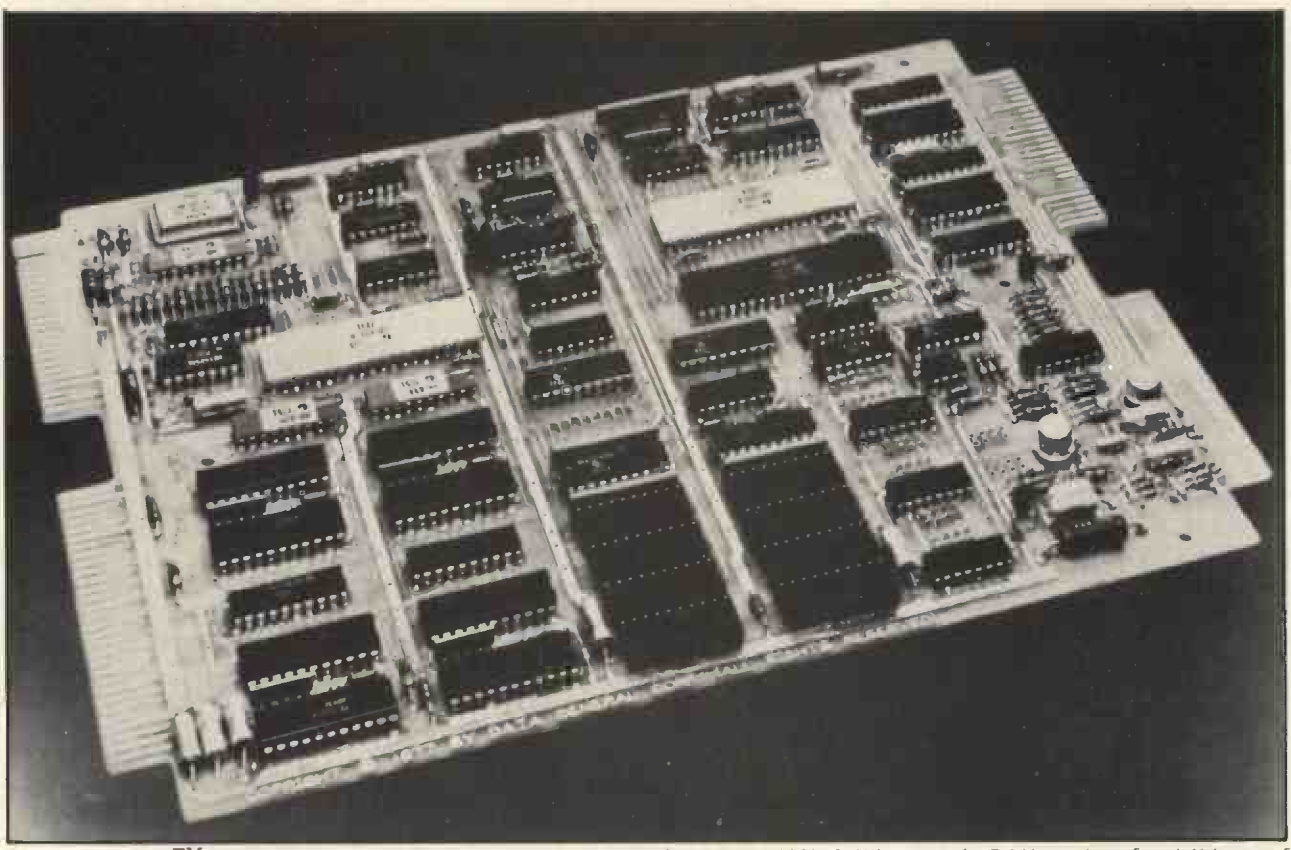
Such synchronous serial transmission is used where human communication is not involved, such as in the communication between tapes and disc controllers.

Intel produce a chip which is a universal synchronous/asynchronous receiver/transmitter (USART). This chip is designed for data communications and is capable of converting the parallel data, required by the MPU, into virtually any serial transmission format presently in use. The device is by necessity programmable and goes under another title of a programmable communications interface.

It is beyond the scope of this article to describe in detail the devices with which the computer can communicate. However, the basic information given here should help in understanding the more specialised articles to be found in PCW.

As Per Your Instructions...
UNDERSTANDING INSTRUCTIONS & DATA CODING

Victor Nicola



The microNOVA™ Board Computer MBC/1 combines a 16-bit microNOVA CPU, 2 K-byte static RAM, sockets for 4 K-bytes of PROM, an asynch interface, and a 32-line digital I/O port on a single 7½" x 9½" board.

A computer, since it operates automatically, must be supplied in advance with both instructions and all the numerical data it needs for the complete calculation. The schedule of instructions, the program, will be stored in the computer's main memory together with the data in the same way: if one examines the contents of a certain location in memory at random it is impossible to tell if the contents of this location are an instruction or data. The representation of the instruction in this form is known as coding.

Coding is nothing more and nothing less than a *pattern* of 0's and 1's (bits) next to each other. Each pattern means something to the computer, according to its design. The actual coding of instructions is done either by the programmer if he wishes to write his program in *machine language* or by the compiler, or interpreter, if he wishes to use a high level language (e.g. FORTRAN or BASIC). The computer, when it comes to executing the instruction, will decode the contents of this location and act accordingly. (The locations containing *data* will not be accessed for fetching an instruction but will be accessed *by* the instructions to fetch the data for the purpose of manipulation or comparison).

The coding of an instruction is done according to the *instruction format* pertinent to the computer in use and, in general, a coded word will be interpreted (decoded) differently in different computers. However, there are computers (or microprocessors) that were designed to be software compatible with computers already in the market and, in this case, the instruction formats are similar and the instructions are decoded in identical fashion.

In Data General's microNOVA computer all instructions are 16 bits wide and the instruction format is, broadly, as follows:

Type 1

Bit number	00	01 02 03 04	05 06 07	08 09 10 11 12 13 14 15
Bit pattern	1		Op Code	

If bits 0 of the instruction word is a 1 then the instruction is of the ARITHMETIC AND LOGICAL type – the computer will look at other bits in the instruction word to tell it what *registers* or *accumulators* are to be manipulated, any modification to them to be made and if the next instruction is going to be executed or skipped. The *type* of manipulation, the *Op Code*, must *always* be part of the instruction and, in this case it is bits 5 - 6 - 7.

Type 2

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	1	1			Op Code										

If bits 0 - 1 - 2 are 0 - 1 - 1 the computer will decode the instruction as an Input/Output instruction. Other parts of the instruction word will determine the I/O device to be communicated with and what type of communication: send or receive data, test a certain element in the device, etc.

Type 3

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	0	0	Op Code												

This type of instruction will be decoded as referring to *another* memory location for the purpose of modifying its contents or for fetching the next instruction to be executed from it. For example, a jump.

Type 4

The last type: bit 0 = 0 and bits 1-2 are either 0-1 or 1-0.

This type of instruction will be decoded as an instruction which references memory for the purpose of fetching data from it or storing data in it. The data fetched from memory during the execution of these instructions is then manipulated according to the instructions that *follow*, and then stored back in memory for future use; or sent to an I/O device; or it helps to determine the address of the following instruction(s) to be fetched from memory (indexed addressing).

So much for a brief idea of the encoding of instructions. Now, how about the way *data* is encoded?

The data is stored in memory is, also, encoded in a certain fashion which the program recognises and is able to decode. Data used in arithmetic operations is normally represented (encoded) in a form, known as 2's complement, which simplifies the arithmetic manipulation of these numbers in addition and subtraction, regardless of their sign: addition or subtraction of numbers represented in this form (code) results in a number represented in this form and no provision is required for testing the signs of the numb[illegible]ior to the actual addition or subtraction.

Before *out-putting* the contents of a memory location or a CPU register to a printing device or to a VDU terminal the data must be converted (coded) to (in) a form recognized by the I/O device; the most common code used in terminals and hard-copy devices is ASCII. This code provides a unique binary representation (7 bits wide) of every alpha-numeric character and, also, special symbols (e.g. ?,@,£,=,etc.).

The conversion is done first by calculating the *decimal equivalent* (including sign) of the binary (2's complemented) number to be output and then coding every digit and sign of this number in 7-bit ASCII; and only then output the number and any other symbols required to the I/O device in this order: sign, most significant digit, . . ., least significant digit, preceded and/or followed by any symbol required.

Similarly, characters *input* from a computer terminal will normally be in ASCII code and they will need to be decoded by the program to interpret their significance. For example, if the characters input are to form a number the program has to ensure that they are numeric characters and if not, reject them prior to decoding them; or, if they are valid, code them in internal coded form.

ASCII (American Standard Code for Information Interchange) is a code that was originally devised for serial communications to supersede Morse code. Its advantage over Morse code is that all its characters are of fixed length (seven bits) while the Morse code is of variable length.

Historically, the first computer terminals were Teletypes (TM) which were originally designed for communications purposes. This code is still widely used although other codes have been devised since (e.g. EBCDIC) and are in use. Teletypes (TM) normally transmit and receive eight bits, the eighth bit being either "padding" or a parity bit. A parity bit transmitted with the character enables the receiver to determine if a bit has been corrupted during transmission. For example, the character A is transmitted in the form 01000001 if even parity is used, or 11000001 if the first bit is "padded" with 1.

To conclude, data transferred from one part of the computer system to another must be coded in a fashion which the receiving end can decode. The coding is normally done in industry standard code if the data is transferred between the main computer and the I/O (e.g. ASCII) or in a special code if it is transferred within the computer (e.g. instructions and data).

Things you wanted to know about micro's but were afraid to ask

Mike Dennis

HINTS ON SYSTEM DESIGN

There haven't been many articles written on how to get started on designing your own system. Circuits have been given for specific systems e.g. 77-68 but no general guide lines for the intrepid venturer who wants to do his own thing. Well, here they are!

The main points to consider (but not in any particular order) are:—

1) System expansion
2) Availability of programs
3) Range & cost of peripheral chips
4) Availability of published system circuit diagrams
5) Other technical data
6) Availability of monitor programs.

All six points should be considered when choosing which CPU chip to use and when planning your system.

System expansion

Everybody would like to have the whole gamut of peripherals from floppy discs to printers. However, the majority of these devices have relatively simple interfacing requirements and thus do not significantly affect our basic design considerations. The only real exception is the floppy disc which, in general, has a rather sophisticated software backup which is often called *DOS* or Disc Operating System. This software resides in memory and often requires at least 4 or 5k. This conveniently brings us to memory expansion and it is this that the majority of people will want to consider first.

A very basic design will use only 256 bytes of RAM. My advice is to initially go for *at least* 1k bytes for the initial system unless you are really impecunious. The approximate cost for 1k(bytes) using 2102's is about £10. This is just the bare cost for the devices and doesn't include any sockets, buffers, special offers or purchase from the States! You will probably have heard about tri-state buffers and if you are thinking of expanding your system past about a couple of kilobytes then you really ought to consider buffering. Nothing traumatic about this.

The maximum per kbyte that you will need are two octal tri-state buffers e.g. 81LS95 or 81LS97 and will cost you £1.50 for the pair. Some buffers have a small amount of gating on the Enable lines which can be very useful and save you a gate or two. For this reason, the 81LS95 is recommended but you can buy buffers other than the types that I have mentioned. They have different characteristics particularly the amount of drive that they can push down the bus. I suggest that the drive capabilities of the 91LS95/97 series will be more than adequate for your system. Buffering the addresses where they come into each RAM board is a luxury at this stage which you don't really need. If the RAM that you use has separate inputs and outputs then you may be able to get away with just one buffer on the outputs of the RAM — see Fig. 1.

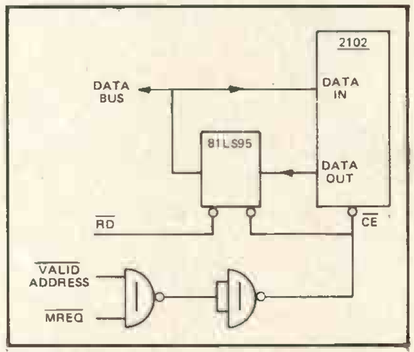


Figure 1

Once you start buffering, though, you have to go the whole hog and buffer the world, CPU and all!

Cost

You will probably want to expand your available memory and assuming that cost is your main criterion then you can go a number of different ways. The most logical step up is a 4K block (no pun intended). You can, of course, carry on using 2102's and a 4K block will set you back about £40, take 32 sockets, need an awful lot of wiring and about 1¼ Amps from your power supply. A more elegant solution, slightly more costly (about £45 from the States), are 2114's which are 1k x 4 and so you need two chips per Kbyte making a total of 8 chips for 4K. Power consumption is still fairly high at about 800mA for the lot.

One advantage of using 1K x 1's or 1K x 4's is that this particular format lends itself readily to write protection. With a little bit of logic and some switches you can easily write protect the 4k block for any amount using between 1 and 4k in steps of 1k. A suggested circuit is given in Figure 2.

I've mentioned sockets and their use is debatable. Certainly, I would recommend their use for the larger chips like the CPU, UART, ROM etc. Is it worth using them for RAM? Well, if you are using basic RAM chips like 2102's, the actual cost of the sockets will be quite a high proportion relative to the cost of each chip. With chips such as the 2114 then relatively their cost drops as you only need 8 sockets for 4K bytes.

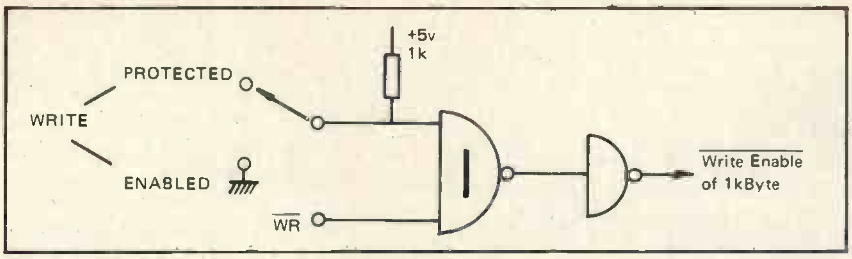


Figure 2

I would suggest then that you use sockets for the latter type of RAM but if you are going to use lots of 2102's then just buy 8 sockets, write a short memory testing program, test your RAM as you buy it and then solder them in directly.

Meanwhile, back at the ram ...er, ranch. Can we do any better than 2114's? Well, the answer is yes. Both previous types of memory were *static* in operation. Once we have written the data in, it will stay there until we turn the power off.

Another type of memory is *dynamic* in operation and here the data is held as a small charge in tiny capacitors. Unfortunately, the laws of physics being what they are, this charge will leak away unless we periodically recharge or *refresh* the chip. One such chip is the 4027 (4k x 1) and we will need 8 chips to get our 4Kbyte for about £16 – about one third that of the static. Power consumption is low (worst case – about 200mA total) and we have the additional advantage of a standby facility on the chip where the power consumption is virtually negligible. Well, you don't get something for nothing so with this particular chip you need three supplies (+5, +12 and –5v). However, most systems will have these supplies already available and so this should be only a minor disadvantage. The major disadvantage is the circuitry required to refresh these RAM's.

What exactly happens during refresh? Basically, we have to generate certain signals to cycle through all of the memory locations within a certain time – usually 2mS. The exact signals will depend on the type of RAM chip used. We need some form of addressing circuitry to sequence through all the locations. It is here that the Z-80 reigns supreme as all of this refresh addressing or cycling is generated automatically within the CPU and is available for our use without slowing down the execution speed of the CPU – what is called *transparent* operation. You can build other refresh circuits but they are at best a bit of a bodge (American jargon for this is "kluge") and will cost still more money when compared with the Z-80. So if you want to use cheap memory, go dynamic and if you have no other preference, then choose the Z-80.

Availability of programs

The 8080 and 6800 were two of the earliest and thus most popular CPU's. As you would expect, there are probably far more programs written for these two than for all the rest put together. This shouldn't be taken to mean that they are better CPU's – just the first.

Does the availability of programs really matter? I don't think so, for the following reasons. I see programs as falling into three categories. Fairly simple games etc., that require the minimum of modification to work on your system but will probably soon lose their initial excitement; more complicated programs e.g. mathematical sub-routines, which will require larger memory than you probably have at the moment and may require fairly extensive modifications; and finally, the utility type of program such as assemblers and editors which will almost certainly require yet more RAM and may or may not require very much modification. However, if you are the software type, and this article is really aimed at you, then I would suggest that it doesn't really matter that all these programs exist as you'll soon be writing your own. If, on the other hand, you don't want the bother of writing programs then you probably won't want the bother of building your own system either.

If program availability is of interest to you then don't forget that the Z80 will run 8080 code. The SC/MP doesn't really appear to have as yet caught the imagination of the personal computerist and so there isn't very much software appearing for it these days. (PCW But see the "Micro Muse" articles as well as "Super Scamp" in a coming issue). Another newcomer is the 6502 which is similar to the 6800 in specification and is bus compatible. It also has the advantage of an on-chip clock which was always one of the potential bug-bears of the 6800.

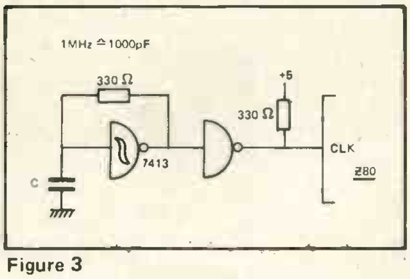


Figure 3

The clock requirements for the SC/MP and Z80 are both very simple and a suitable driver for the Z80 is shown in Fig. 3. The oscillator frequency can be set by adjusting the value of C.

The 8080 needs fairly complex clock circuitry. National do make a suitable clock driver chip that does all this but when you consider the fact that the 8080 also needs another special chip to latch out the status bits, then it really must be considered as a three chip CPU.

The 6502 is used in the KIM, PET and lately the Rockwell and so there is quite a lot of software available for it; particularly for the PET. However, don't forget that software can be very hardware dependent and so be wary of buying any program without any object code listing as you may have the devil's own job trying to patch it into your own system.

Monitor programs

These are the 'executives'. Some of the micro's have in-house monitor ROMS available with monitor programs already in – such as MIKBUG, and more latterly SWATBUG for the 6800 and KITBUG for the SC/MP. This takes all the hassle out of starting up since the ROM provides all the basic house-keeping and input/output routines to communicate with the world at large. I haven't heard of any that are readily available for the 6502 and certainly there aren't any for the 8080 or Z80.

Computing without some sort of resident monitor can be appalling! You talk to the computer at toggle switch and LED level and very soon your painstakingly entered program disappears in a flurry of Stack Pointer as it comes charging down through memory eating program as it goes. There is nothing wrong with this method and it is the cheapest way to get computing. However, you will soon either get very frustrated or else you'll get bored by the whole thing and take up macramé. You really are stuck without a monitor ROM.

There are solutions, however, which are not all that painful. You can store the program on tape, toggle in a bootstrap loader program and then load in the monitor. However, unless it is in ROM or write-protected RAM, you still stand a pretty good chance of wiping it out. So you really need a monitor in ROM.

Having decided to use a monitor, you then have the problem of how to get one. You could use Neil Harrison's excellent MINMON that appeared in these pages or get one of the monitor programmes from the various libraries that the ACC runs. Having chosen your monitor, you still have the problem of getting it programmed into ROM as you need access to a PROM programmer. I can supply a 2708 ROM programmed with the latest MINMON already in it for £17.50 or alternatively program your own 2708 for £10.

If you're thinking of writing your own monitor program then get it debugged first before you try and commit it to ROM otherwise you know whose law will prevail.

Cost of chips

The prices for the various CPU chips vary considerably. One particular advert quotes the following:

Z80	£15.50
6800	£11.25
SC/MP	£10.30
6502	£15.00
8080	Not quoted.

Frankly, unless you're given an 8080, I wouldn't recommend anyone buying it. Various micro's offer different speed versions but again unless you're really hooked on speed (and I would suggest that to quicken things up you should look to your software) then don't bother with the higher speed versions as they are more expensive, need faster memory and peripheral chips. You could try a mix n'match of say the Z80 4MHz version with 2MHz PIO and CTC and incorporate a wait state to slow things down just a bit. The overall speed will be faster than the 2MHz version.

The Z80 and SC/MP will give you the advantage of static clock operation which means that you can hang LED's on the address bus and data bus and single step through (*each* clock cycle should you wish) and see the Stack Pointer appear on the address bus as you return from a subroutine. This can be a useful tool for really nitty-gritty debugging especially if you get stuck in a program loop and never come out. Be warned – if you do this on the Z80 with dynamic RAM in the system then this technique is guaranteed to wipe out your data since automatic refresh doth cease. For the same reason, if you use this CPU, then use some form of monostable on the reset line to the CPU and don't hold the WAIT line low for too long. The SC/MP doesn't have these restrictions but neither does it provide for refreshing!

As you can see, the price of the CPU chips are much of a muchness. You are almost certainly able to find them at cheaper prices than those I have quoted. My advice is don't look at the cost of the chip but look instead at the spec, what it provides and whether or not it does all that you want. The extra price of the Z80 is far outweighed by its 'many advantages'!

Range of chips

The range of peripheral chips is legion. Most CPU manufacturers provide some form of parallel interface chip – be it called PIA (6800) or PIO (Z80) or whatever, the basic function is the same. The cost of the Z80 PIO is almost double the Motorola PIA and both have their advantages and disadvantages.

The main advantage of the Z80 PIO is the ability to respond to a preset (programmed) combination of signals on one of its' ports. This saves you having to write a program to periodically sample the status of the port to see whether or not the specified condition has been met – a technique called polling. The PIO does that bit for you and responds by generating an interrupt. The Z80 PIO is also rather more sophisticated in its' response to interrupts as it can supply part of the address for the interrupt sub-routine.

However, for the beginner, I would suggest that interrupts are only of limited use although it's good to have these more advanced facilities in reserve. The main advantage of the 6800 PIA is the ease with which the states of certain control lines can be set by the program. The recent PROM programmer design by Dave Goadby in a recent issue of PCW used this to good effect.

Another chip is the UART type or Asynchronous Communications Interface Adaptor or ACIA as Motorola call it. All this chip really does is done in two halves. One half converts parallel bytes to serial format and the other half, vice versa. The Motorola part is called the 6850 and is quite good, uses less pins then a conventional UART, can interface to a Z80 and is cheap. Forget about the Z80 SIO as it is very pricey.

There are many other types of peripheral chips available but which are too numerous to mention. Perhaps I ought to mention in passing the Z80 CTC or Counter Timer Chip which has four timing channels which can be quite useful at times.

If you are just starting up avoid DMA controller chips and CRT controller chips like the plaque unless you have a tried and tested circuit to work to. However, seeing a circuit in print in the pages of a magazine doesn't mean it will work. Quite a lot of people who tried to build a VDU that was published in one of the other magazines some time ago will testify to that fact. Even PCW is not entirely blameless though we do try – honest.

Memory mapped VDU's are good and allow much greater flexibility in writing programs that involve graphics and an interactive response. There have been various circuits around but be a bit wary of any using greater than 64 characters a line because if you are going to use a normal television, then you may well find the characters looking rather blurred. Even 40 characters a line look pretty awful on some sets!

Availability of published complete system circuit diagrams

There is the 77-68 budget system based on the 6800 and marketed by Newbear. It's a basic system but expandable which is important. There have hardly been any designs for a Z80 based system (I really must get round to doing it). If anyone out there has a good design for any other CPU then please let us know.

Other technical data

The Z80 Technical manual is a must and very readable but compare their programming manual with Osborne's "Z80 Programming for Logic Design" before committing any hard cash. The Motorola Applications manual is a gold mine for the 6800 but again think carefully about their programming manual. Have a look at Osborne's book first.

Other good sources are the Texas TTL Designers handbook which again is a must. You can get it from A. Marshalls of Edgware Rd, and Cricklewood Broadway or from any Texas distributor. The cost is about £5 and is about the same as for other data books from other manufacturers such as National and Intel.

Ideally, you need everybody's data books since all the manufacturers make odd useful i.c.'s that aren't commonly known. However, that would be rather an overkill so I recommend that you buy the Texas book and perhaps one other plus, of course, the Technical manual for your chosen CPU chip. RCA have excellent literature on their 1802 micro.

Well, that's about it. Stop thinking about exotica like the 6809, 8086 and Z8000 and go out and buy something. There will always be a new and better micro 'just around the corner' and which will always be cheaper 'next month', as too will the RAM that it uses. If you follow this path of reason then you will never get started.

And here I would like to take the opportunity to apologise to the more knowledgeable among you for the rather sweeping generalisations that I have made from time to time.

However, in the context of the article I consider them to be valid and hope that I've given at least some of you a few pointers to help you along the way. My address for the PROM's is:

The Corner House,
Birlingham,
Pershore,
Worcs.
Tel: Evesham 750251

but please – no personal callers.

BYTE BASHER'S LIFE SAVER

Brian Hewart, *Hewart Microelectronics*

LIFESAVER

First Aid

A dead or half dead micro is one of the most frustrating pieces of electronic equipment you will ever have to work on, and yet the most common faults can be traced by the simplest of equipment. Here is a circuit of a logic probe that can be built in a few minutes on a piece of veroboard 15mm by 120mm, with the tracks running across the short side. It is not a particularly sophisticated probe, it can't tell the difference between an open circuit and a logic 1, but it will find most of the faults that kill off micros.

Build it any way to suit yourself, but I suggest that you start by mounting a thick piece of copper wire as a probe at one end (the earth wire from mains twin and earth is fine) and fix the 2 LEDs next to it so that they are in view as you touch the probe onto a test point. The rest of the components can be fitted further up the veroboard with two power leads fitted with croc clips at the far end from the probe.

When you have built the probe, test it by touching the tip to ground, the Green LED will go off and when you remove it from ground the Red LED will flash. Note that this probe presents a 1 TTL load to the circuit. If you use it on a low fan out circuit such as an unbuffered MOS circuit you may not get reliable results. There is no LS121 available but you could add a 74LS04 and use 2 gates as a non inverting buffer if your application requires a higher input impedance. But generally speaking, unless you have a dedicated system doing one job only, it is good practice for future expansion to have some TTL load drive capacity spare in any system.

Debugging track shorts

Now let's look at how to debug a track to track short on a Motorola 6800 based system. This is the most common fault that you will come across and I admit without too much shame that I have not yet built a system without one short. When you reset a 6800, the address lines go to FFFE which is 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 in binary, with the 0 being address bit zero. If you tie the reset to ground and run the probe along the address bus you should get the above set of readings. If you don't, trace out any that are wrong, looking for a short. You will be very lucky to find a fault this simply and it would not find two address lines shorted together (except 1 & 0) because they are all logic one.

Next stage is to remove the 6800. This should create all 1s on the address and data busses. Get a jumper lead clipped to ground and one by one ground the busses, in turn testing; 1, the far end of the bus for ground; 2, the two tracks on each side of the bus. Work your way through all the tracks and usually you will come across two shorted together.

If you cannot find a simple two track short you really are in trouble. It gets very messy from here and those of you with a nervous disposition should switch off now. We have got to the stage where we are running out of general tests because all systems vary, so from now on we will continue on the lines of a Mikbug orientated system. We have created address FFFF by pulling the 6800. At FFFF the system expects to find ROM with the address of the start of the monitor. In Mikbug this is EODO, so we should find DO at FFFF. Look for DO on the data bus, this is 1 1 0 1 0 0 0 0 in binary. If you do not get DO on the bus look at the enable pin of the ROM. This is pin 20 on the 2708 type ROM which has an active low input, i.e. this pin should be logic 0 when FFFF is on the address bus, or with the normal Motorola Mikbug ROM in a 6830L7 we have 3 chip selects, pins 10, 11 and 13 all of which must be at logic one. If these logic levels are not found work backwards towards the 6800 and find out why.

When the going gets really rough you get to a point where you have to suspect a faulty integrated circuit. Here there is no easier way than substituting ics but if it is an expensive one you probably do not have a spare. In this case double check your *thinking*. With the ic in place, double check all its inputs and the expected output, this applies to a simple NAND gate just as much as a PROM. When you are convinced that the ic is at fault remove it and try to create a substitute output with jumpers or a current meter. If you get an unexpected rise in current on the whole system or if your meter shows a high current when used to simulate one of the removed ic's outputs then it may be the next ic along that has a faulty input.

To sum up the above static debugging: you can assume that most faults will be a simple two track short or occasionally a break in a track particularly where the holes are drilled. Create a known signal and track it down wherever it should go and make sure it does not stray onto adjacent tracks.

Using life to create more life

If you have a system with a 'bit of life' in it a good test method is the simple loop test. Create a program loop so that the computer is repeating a known set of instructions.

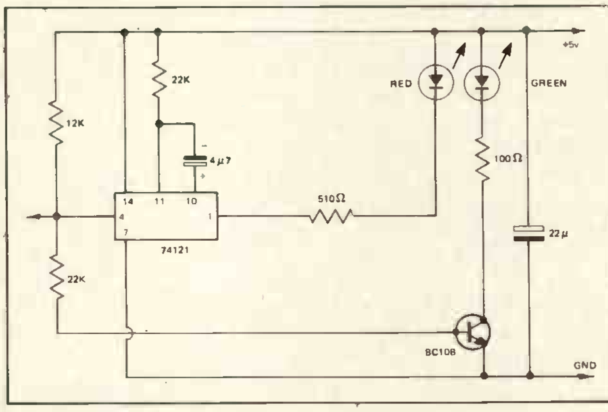
```
A050 B6 00 00     LDA A     Load A from 0000.
A053 20 FB        BRA       Branch to A050
```

Here we have a loop. The A register is loaded from 0000 then the program loops back and does it again until the reset is pressed. Whilst the computer is in this loop it will address A050 to A054 and 0000 in turn and no other address. Also these are all read only instructions, so the Read/Write line stays at logic 1. You can use the probe or a scope to see what is going on. If you find activity anywhere else other than these addresses find out why. Now change the B6 to B7 which is store A at 0000. Now the R/W line will go low when the store instruction is reached. With a 2 beam scope you can put one beam on the R/W and look to see if it is pulling anything else down with it during the Write to 0000.

If your problem is with a UART or similar system, this loop technique is very good for scope testing. You can, say, write a loop to continuously transmit a single byte. AA is very useful because it is alternate 1s and 0s. Trigger your scope from the Transmit Data Strobe or the Register select pin on an ACIA and look at the output.

There are problems with dynamic testing that you cannot spot with a simple probe: clock overlap, slow rise and fall times, ringing etc. If you are going it alone with your own system you should have a scope and be able to use it. If you are building a kit or from a magazine then any reasonable design will not have these problems.

To sum up again, the most sophisticated debugging tool ever invented is the human brain. Use it to work out what *should* be happening, use it to work out what *is* happening and when the two differ use it to work out why.



Byte Basher's Life Saver
Build this circuit on a piece of Veroboard 15mm x 120mm with the tracks running across the 15mm length.
Green ON = logic 1 or Open Circuit.
Red flash = high to low transition.
Steady red = pulse train.

THE MICRO MUSE
Background to Micro-Poetry 2

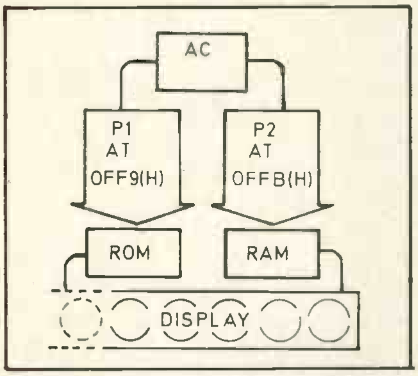
Eric Finlay

1	OF20	C2	
2	OF21	00	
3	OF22	C9	
4	OF23	04	
5	OF24	90	
6	OF25	FA	
	....	...	
7	OFF9	OD	P1 (H)
8	OFFA	00	P1 (L)
9	OFFB	OF	P2 (H)
10	OFFC	40	P2 (L)
	....	.	.
11	OF40	D3	image
		;	end.

The above program puts the alphanumeric— code signal D3 into the centre of the 8-bit LED display of an **SC/MP 11 development system.**

Before we can design such a program we must have a clear idea of how the microprocessor is equipped to carry out such a task.

Consider the following diagram:—



Three registers, two areas of memory, and the display are shown. AC is the accumulator register which handles most of the data used in a program, since it is used for data transfer, and in arithmetic and logic operations. P1 and P2 are Pointer-registers, these can be programmed to point to any address in ROM (the control program of the system, stored in Read Only Memory), and in RAM (Random Access Memory) which holds most of the data for a new program. In order to light up the fifth display digit from the right with the data D3 (translated into an alphanumeric sign), we have to put the D3 data into RAM at a specific address (OF40). This is done at step 11 in the above program.

Next, we have to point one of the Pointer-registers (P2) to the address in RAM at which we have chosen to store the data D3. This is done in two stages, steps 9 and 10 of the program. 'H' and 'L' signify the High and Low digits of the chosen address in RAM. OF is High, and 40 is Low. The Pointer-registers have constant addresses in the control program, which directs the Central Processing Unit (CPU), and these addresses, for P1 and P2, are as shown in the program.

We now have to point P1 to the first address in the control program in ROM which tells the CPU which digits in the display are to be activated. The address of the first right hand digit of the display is ODOO. To put our signal in the middle of the display we add a 'displacement' instruction to the program (Step 4). To point P1 to address ODOO in ROM, we put the High and Low digits c: the address into the two P1 addresses. which are shown in steps 7 and 8 of the program.

The computer now contains almost all of the information required to be able to carry out our program, which we must now design.

Step 1 uses the instruction CO(LD)Load, to load the data at OF40 in RAM into the Accumulator by way of the P2 register. We change the instruction to C2(LD) so that the CPU knows we refer to P2.

Step 2 of the program tells the CPU that it is the OF40 address in RAM which is used. The second address in a possible 'data stack' constructed from this address, would be OF41, which would be displaced from the first address by 01. The next address in the stack would be displaced by 02, and so on.

When the first two instructions of the program are carried out by the CPU, the information D3 will be transferred from address OF40, to the Accumulator register.

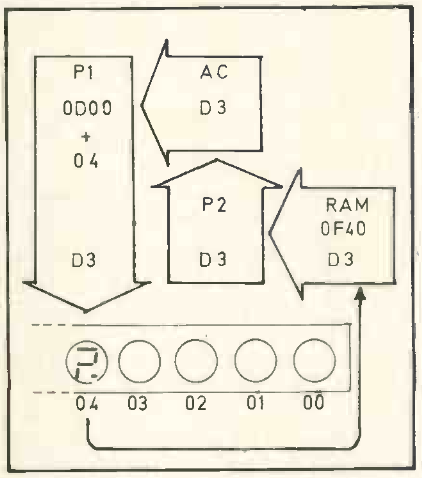
We use the C8(ST) Store instruction to put our information into the display by way of Pointer register 1. To show the CPU that we refer to P1 we add 1 to the instruction. Step 3 of the program therefore becomes C9 (ST) Store. Step 4, as discussed above, tells the CPU that the address to which the P1 register points, in the Display Routine, is to be displaced by 04 to bring our information into the middle of the display, so we put that displacement data into program address OF23, as shown.

If the program stopped here, our information would light up the 04 position in the display for an extremely brief time, since the CPU operates at a radio frequency (14 Kilohertz approx.). To sustain the image in the display so that we can examine it for as long as we choose, we use the next two steps in the program to form a loop, which repeats all the operations of the program in a continuous cycle.

Step 5 in the program is the unconditional Jump instruction 90. However, we have to tell the CPU whether we want it to jump forward or backwards in our program, and we also have to tell it how far to jump. Positive hexadecimal numbers tell the CPU to jump forward, and negative hexadecimal numbers tell it to jump back.

The beginning of the programme is 6 steps away from the Jump data instruction, and so we put FA, which is hexadecimal for −6, into address OF25 of our program.

We have now completed all the programming steps necessary to get the microprocessor to obey our instructions. A second diagram will show how the program circulates our data, and lights up the display at position 04.



By changing the image code at OF40 through all the possible hexadecimal numbers from 00 to FF, we can write all the possible 'poems' that can be written for a single L.E.D. display unit. We can save wear and tear on the muse by getting the computer to perform this lengthy task for us. The following program tells the computer to perform all the possible variations on 2 bit hexadecimal numbers. I will leave the reader to work out how it is done.

OF20	8F	
OF21	FF	
OF22	8F	
OF23	FF	
OF24	8F	
OF25	FF	
OF26	8F	
OF27	FF	time delay
OF28	A8	
OF29	70	
OF2A	C9	
OF2B	04	
OF2C	90	
OF2D	F2	jump −14
.....	..	
OFF9	OD	P1 (H)
OFFA	00	P1 (L)
0000	00	; end.

In my next article I will give a simple program for putting the micro-text 6F,50,79,66,40,6D,79,77 into the full 8-digit display. It should be possible to translate this text into English without a computer. The reader is advised that the message consists of two words which form two thirds of a mind-bending pun.

Computers and Art

Brian Reffin Smith

Since the days when computers were made of valves and art was made of oilpaint, people have been trying to forge a union between the two. The results of 'computer art' — a silly term that's hard to avoid — have generally not been well received. The trouble was, of course, that the artists knew nothing about computers, and the computer people knew nothing of art.

So bad poets, musicians and painters produced bad poetry, music and pictures, as they had before: but now with more ballyhoo, faster, and even more mechanistically. Meanwhile every programmer produced nudes, Snoopies, and obscene Santa Clauses. Artists talked cybernetic jargon, and museums grabbed tricksy bits of technology from lab or video display, put frames round them and said, "hey — this is just like art!"

There were, of course, exceptions. Edward Ihnatowicz's 'Senster' was one: giraffe-like mechanical engineering with radar eyes, quadrophonic ears and a computer-controlled, hydraulic nervous system (Fig. 1). It would lower its huge, mechanical head, the girders, valves and tubing clearly visible, towards any gentle movement or sound. A shaken fist or loud cry, however, would drive it away — it would remember you, and not come to you again until the shock wore off.

It was impossible, even having seen it under construction, not to treat it as a living thing. People stayed watching it for hours in the

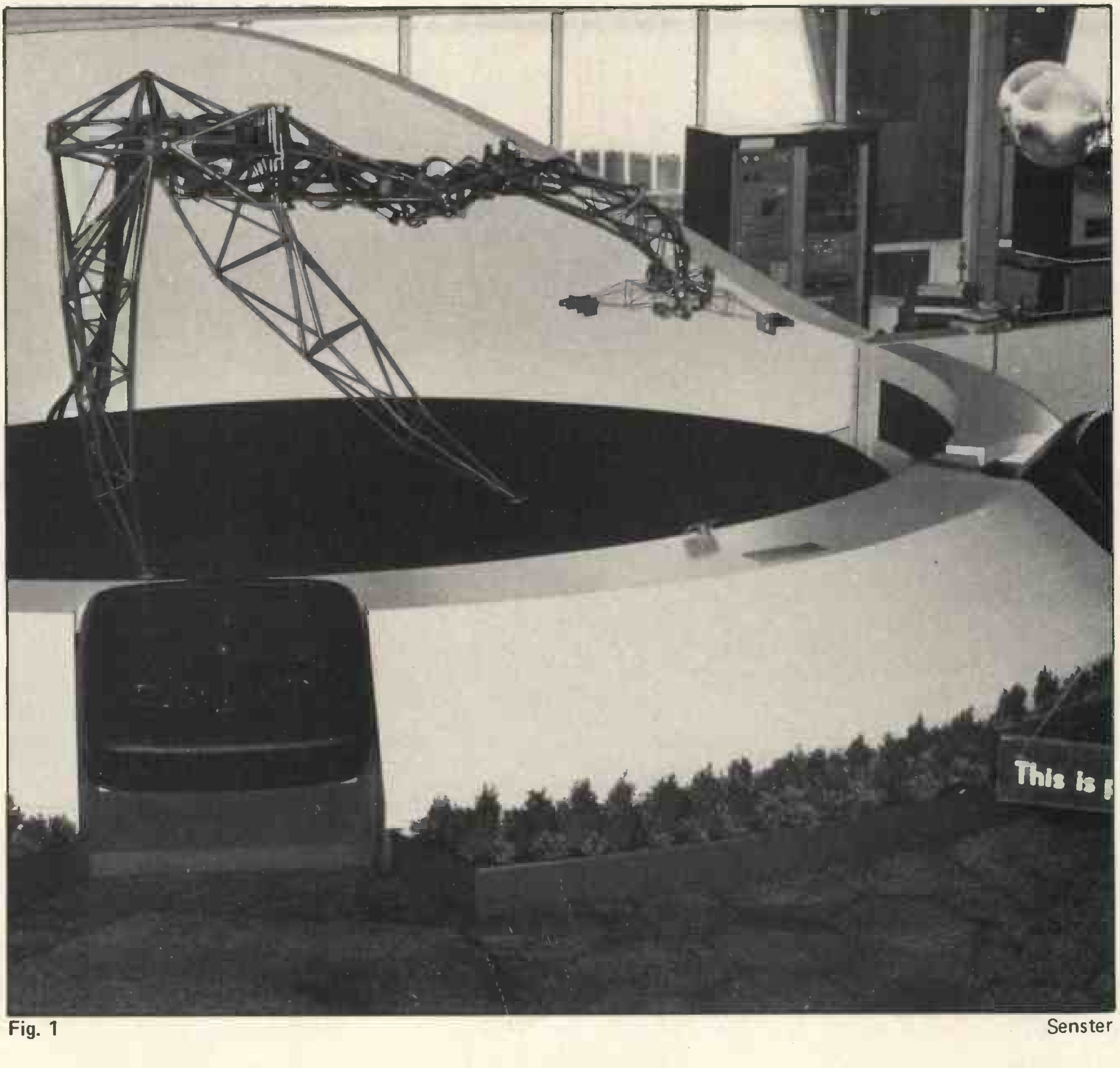


Fig. 1 Senster

Evoluon, Philips' flying-saucer-shaped exhibition in Holland. When birds flew over the glass roof, the Senster twisted up its articulated neck to look at them, like a cat at a window. Eventually, it was sold for scrap. "Too popular", they said. There's a film somewhere. R.I.P. but, Rest In Shameful Oblivion the turgid computer-haiku, the silhouettes of nudes, the whirligig spirals and moiree patterns, the random-number Mondrians and all the do-it-yourself – "we-all-*knew*-art-was-as-easy-as-this" trivia. You've got to start somewhere, but painting by numbers as a career is for robots, not people.

The opposite of the over-eager espousal of computers by some is the near-hysterical suspicion of them by others. In part this is due to the fact that most computers are still doing boring and sometimes dangerous things at the command of boring and sometimes dangerous people. (Personal computers, naturally, are going to change all that – aren't they?)

But the historical roots of the computer/arts chasm (made all the more visible by the bridge-building attempts) are largely accidental. For computers may be said to have come along at the worst possible moment in the twentieth century. Accelerated by the demands of the last war, the computer came into being perhaps twenty years too early. Instead of controlling, for example, processes, systems and pictures, the computer was seen primarily as a number-cruncher. (Based in fact upon logic rather than number, computers are eminently suitable for the handling of qualitative, value-judgemental information, using techniques based, for example, on fuzzy set theory. But we're still told everything's got to be exactly definable.)

Anyway – just when many aspects of social, political and economic life might have been about to crumble, later to mutate into *human*-based institutions, along came the computer, and propped it all up again. We had indeed embarked on the road to an information-based society, but the kind of information and the uses to which it was put, were not at all the result of any radical re-think. Rather, we have systems reminiscent of the Imperial Chinese hierarchic structure whereby the Royal and less-Royal ladies got to have sex with the Emperor.

Though we may assume that technology and hence computers are *in principle* neutral, they of course come to embody the contradictions and power structures of the system that supports them – it is in contrast to this that the ideas of a *liberating* technology and *people's* computers have arisen.

Forgive this slight deviation from the path of computer art *per se*, but it seems essential to be aware of a context for what has been done.

Here then are some questions: now that we've got our computers, what should we do? Now that we can make computer art, what is it for? To emphasise the point, and to explain this long preamble to a piece on computer art:

The idea of using computers in or for art raises some of the most fundamental issues, touching on the validity of the ways in which we spend our money, computer time and lives.

Next time we produce a new computer game, or a scintillating pattern on the screen, let us ask ourselves – "so what?"

It is with this question in mind, and the lack of any good answer for it, that I now somewhat hesitantly approach the immediate topic of ways in which people can make art with computers.

I intend to outline three areas of computer art, going from the relatively simple to the very complex. I shall give examples of others' work or of my own efforts in these fields together with methods the reader might try. I hope also to stimulate ideas as to what might be possible in the future.

The three areas I mentioned above can be classified as follows:

1. where the computer is a tool of the artist, like a paintbrush, printing press or some other means of producing an already totally conceived idea;
2. where the machine's role is to provide areas of detail in some broadly pre-designed plan. An example is the injection of random factors into a design;
3. where the whole composing process is contained in the program, the aim being to see what kind of work emerges from a set of rules or algorithm. Artistic decisions may well take place *inside the computer.* Allied to concepts of Artificial Intelligence (AI), this is perhaps the most interesting area.

In general, as we go from category 1 to 3, complexity and interest increase, while accessibility decreases. Part of the aim of my own work, with 'Intelligent Programs', is to increase this accessibility.

Dealing with the above three classes in turn, we first come to what is for many practitioners and almost all observers, *the* computer art. That is, the computer as a tool. This area is of interest not so much for the resultant work, but because it is still the way in which many computer people attempt to make art.

One might, for example, ask the computer to draw a wavy line such as a sine curve, then repeat it at increasing and then decreasing intervals, giving imitation 'Op-art' effects. Or by combining blocks of characters on screen or paper, produce patterns.

Even here, at this relatively lowly stage, there is a great deal to be found out about the relationship of artistic person to technological machine. With a personal computer especially, you will be limited by the machine. Of course, there may be a million different tricks you can get up to – but how many *categories* of art do they fall into? And what is their validity *as art?* Would the results stand up as art if they'd been done in some other way – or do we just think they're exciting, important etc., because they were done on the computer?

Is it more useful to be able to guide a point of light round a screen, or produce graphic 'characters'? Is colour important? Of course it depends on the user, but how can that user make an informed choice, when he or she has presumably not been able to try a computer for a month or more?

I think it might help to deal with techniques that can be used on any machine, with some small adaptation if necessary. This is artistically more justifiable as well – to give someone the *basis* for doing something rather than too much help (painting by numbers) or too little (purely abstract ideas with no hint at their realisation.)

There have been attempts to provide 'art packages' for computer users, but these have usually been either too trivial or too specialised. I have tried to develop a graphics program for personal computer users that acts, more than anything else, as a trigger for people's own ideas. There follow some extracts based upon it, but here first is a general description.

There are about 20 'commands' of which the user employs one at once. Some of them take a little time to process – but why on earth do we tend to reject routines that don't occur 'instantly'? We're talking about computers, not end-of-the-pier video games. You can draw rectangles, triangles, circles, grids, etc.; and specify or change backgrounds, squeeze or stretch the image, rotate or translate it, and so on. Copies can be stored in memory, reference points displayed, and processes such as random or 'growth' decay initiated. (This latter is a kind of negative LIFE that is applied to the image).

The point is not really to paint shapes on a T.V. screen – why not do it with pencil and paper, if that's all we want? Rather, it is hoped that a person using such a program will dismantle it, shake it up, and do silly things with it, as well as examining some of the relationships between computer methods and art that we touched on earlier. Suppose the commands were computer called, at random (so the 'user' didn't know what would happen next). Suppose you point a film camera at the screen, and change the image frame by frame. Or just throw the system at a group of art students or graphic designers (this is about to happen) and let them get on with it.

The program does, however, highlight one area of difficulty in using personal computers for 'new' things. The areas where some degree of standardisation has occurred (I mean mostly in terms of the version of BASIC used) have in general been those that most readily spring to mind, tending to be numerically oriented rather than, say, string oriented.

So whilst on some machines you can dimension a 20 x 40 string array, like A$ (20,40), and fill each element with just one character, on others you are limited to a maximum dimension-product of 255, or – the most ludicrous limitation – have to declare that all your strings are going to be no more than (but therefore, the machine suspects, no less than) x characters long. So you fill up your memory with potential spaces that you don't need – sounds like a big company trick. Thus a program that will run in 8K on a Research Machines 380Z or PET, would be out of memory on at least one other common machine.

Here, anyway, are some basic routines that should be usable in any machine environment, that are the bases of more complex operations that many readers might invent for themselves. I am sticking here to a 15 x 15 array (less than 255) and one-character strings only.

The 'canvas', then, is A$ (1-15, 1-15) so that the top left corner of screen or paper is 1,1 and the bottom right 15, 15. You can see that what

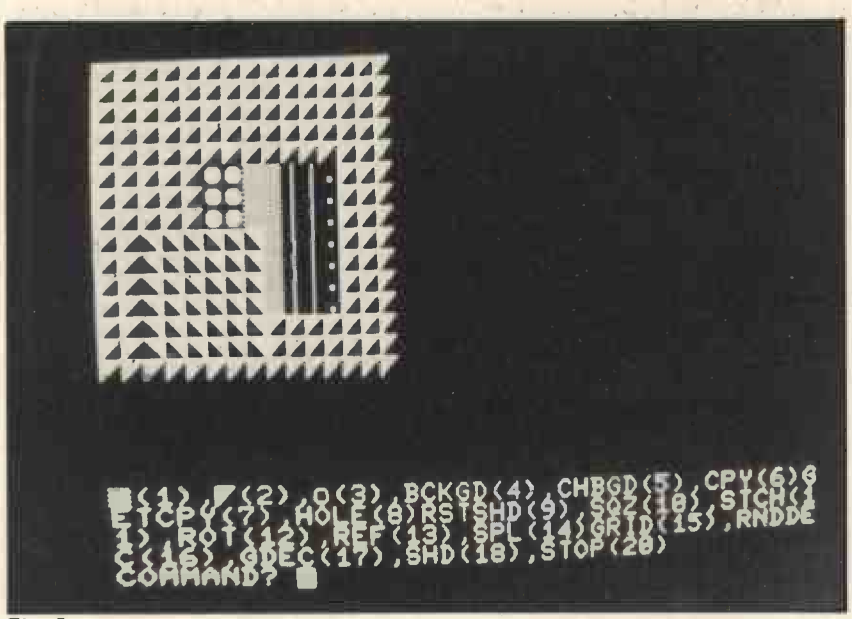


Fig. 2

we have is now equivalent to a piece of graph paper – on which one can plot, fill elements systematically and so on. The following will enter a rectangle of stars onto the 'canvas':

```
100  PRINT "ENTER CO-ORDINATES
     OF TOP LEFT CORNER"
110  INPUT X,Y
120  PRINT "ENTER HEIGHT AND
     WIDTH"
130  INPUT A,B
140  FOR I = X TO X + A
150      FOR J = Y TO Y + B
160          A$(I,J) = "*"
170      NEXT J
180  NEXT I
```

Then this will fill in the background with "/"s:

```
200  FOR I = 1 TO 15
210       FOR J = 1 TO 15
220          IF A$(I,J) = " " THEN
                       A$(I,J) = "/"
230       NEXT J
240  NEXT I
```

To print the result:

```
300  FOR I = 1 TO 15
310       FOR J = 1 TO 15
320           PRINT A$(I,J);
330       NEXT J
340       PRINT
350  NEXT I
```

To change the background, just read each element of A$ and if it is "/" then change it to, say, "@". You can also put in rectangular 'holes' into the image by making the symbol for the actual shape a space, " ".

To obtain more complex shapes such as triangles requires a little geometry, but the graphics program avoids even this small amount where possible – a circle for example is built up by approximations to the shape rather than by the formula – which itself would need a great deal of use of the 'INT' function to provide whole number outputs for A$.

Many effects can be achieved by taking a piece of squared paper, drawing out the shape and then deducing the rule(s) that the computer would need to reproduce it. Examples are grids, spirals, even mazes. Imagine again the A$ 'canvas', with 'I' increasing vertically downwards, and 'J' from left to right. To draw a spiral with a one-cell gap between the actual trace, the array is filled in with alternate I's and J's, each pair alternately increasing and decreasing, and incrementing by 1. So given a starting co-ordinate of x, y, J increases by 1, I by 2, then J decreases by 3, I by 4, J goes up by 5, I by 6, then J goes down by 7, I by 8 and so on – test for the edge of your canvas, when I or J are 1 or 15.

Rather than reproduce bits of listing which may, out of context, mystify rather than help the reader, here are a few algorithms in words for achieving a few more operations:

Rotate: input centre of rotation, and angle (convert to radians – multiply by pi/180). Then for each old point, the new co-ordinates are given by I*cos A – J*sin A for one axis, and J*cos A + I*sin A for the other.

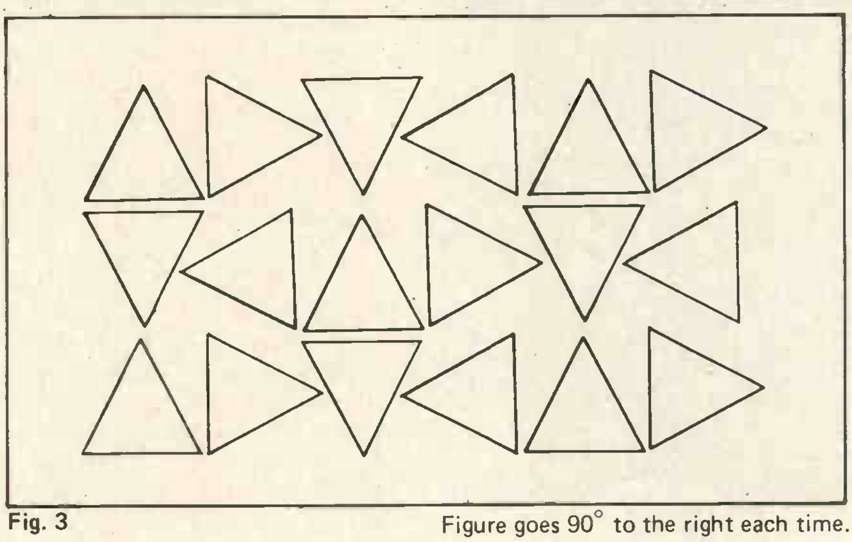


Fig. 3

Figure goes 90° to the right each time.

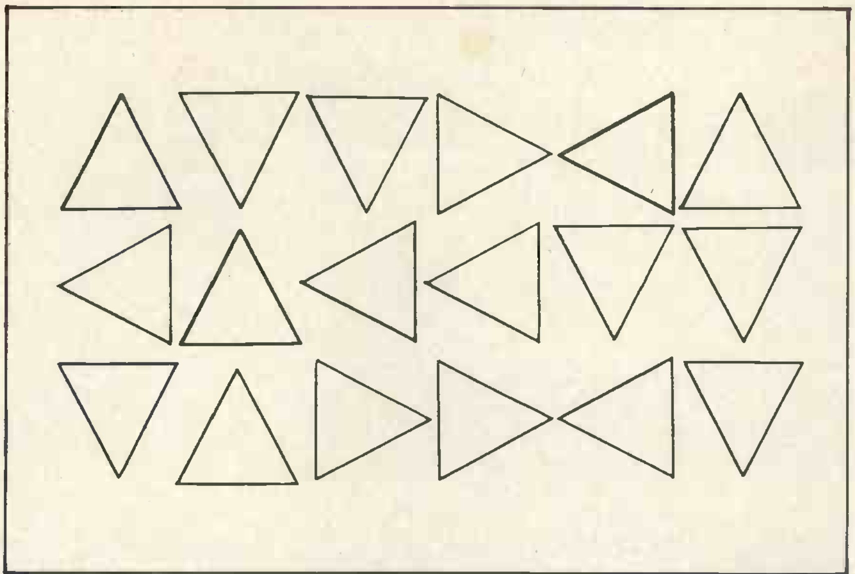


Fig. 4 Random rotations

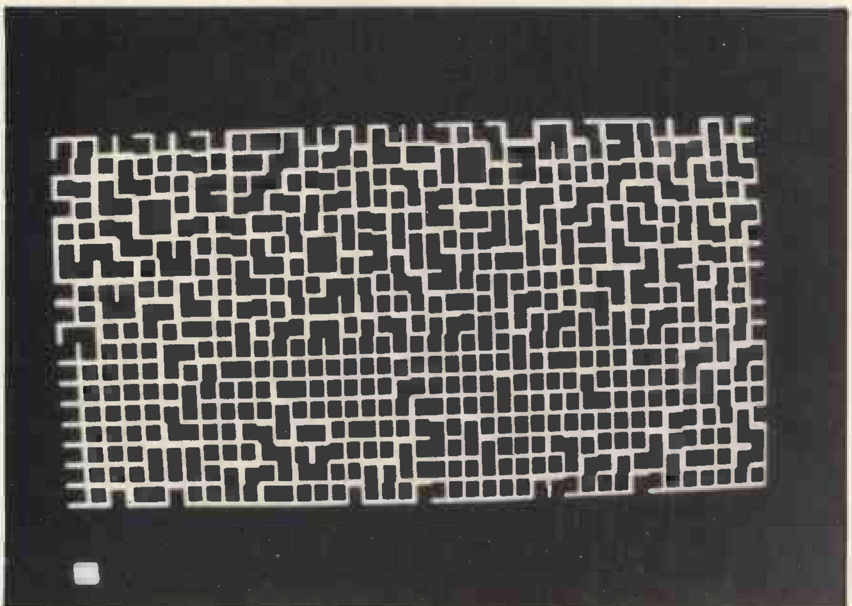


Fig. 5

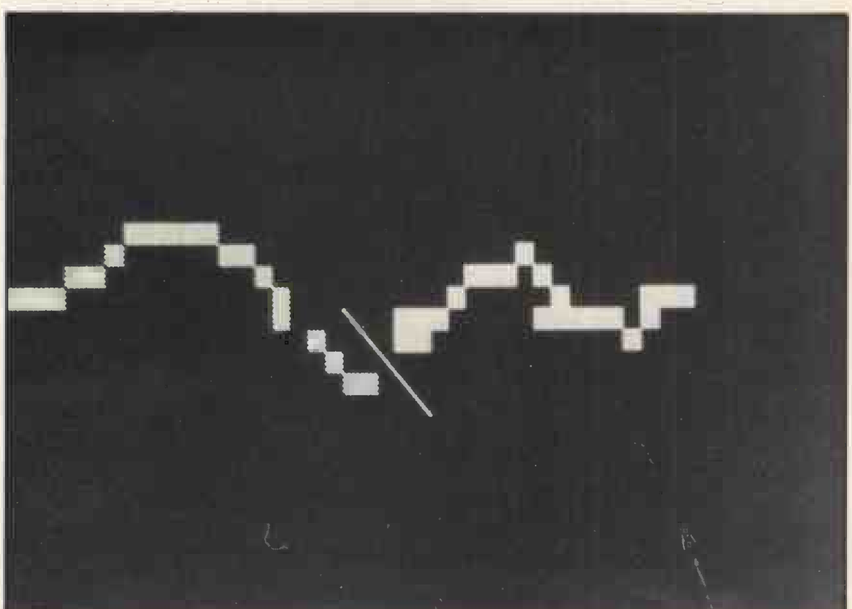


Fig. 6

Decay: For random decay, get pairs of random nos. between 1 & 15, say 30 at a time, and make that element of A$ the decay symbol (space, etc.). Repeat three or four times. For growth decay, put in at least one 'seed' of space, and allow it and any others to take over any adjacent elements, subject to whatever rules you like.

Stretch: take a chunk out of the middle of the picture, say I= 5 to 10, J = 1 to 15, expand and copy it so that it fills up the whole space. Thus row 5 goes into 1 and 2, 6 into 3 and 4, 7 into 5 and 6 and so on. When you get half way, start putting 10 into 15 and 14, 9 into 13 and 12 etc.

Squeeze: the converse of this – miss out alternate lines and copy the rest into the middle 50% (or whatever).

Shading: Read into say, S$ (1 to 4) either graphic or alpha-numeric characters going from light to dark. Then, as you build up an image line by line, calculate each line's length, divide by 4, make the answer an integer, and that gives you the number of times each shade character is to be repeated in that line, with maybe a few left over. e.g., a line nine elements long is shaded S$ (1, 1, 2, 2, 3, 3, 4, 4, 4)

An example of the sort of image produced is shown here (Fig. 2) but everyone will be able to think of some new trick to incorporate. You can see that in terms of the three areas described earlier, even this simple program crosses over from the first to the second.

Of course the resolution is lousy – but then we are presumably not trying to obtain engineering drawings, while for graph plotting routines there are better ways of doing things. To reiterate, I am not concerned so much with the quality of the images, as with the ideas that are stimulated – the 'what if . . .' questions.

To bridge the gap to the second of my three categories, where the computer takes over some of the final design, which however is still broadly outlined by the person . . . consider some simple process such as reproducing a shape a number of times. Telling the computer to turn it 90° to the right each time will produce a fairly predictable result, while using a random process will not (Figs. 3, 4). Or we could feed words into the computer and tell it to give us random sentences made up of them – not that easy, since some thought must go into choosing the initial list of words if you want the output to mean something quite often. An example of output:

> clouds golden. i wandered once saw golden lonely host. lonely cloud daffodils of golden when i saw. cloud golden crowd lonely.

But now consider applying one or two *rules* to the process. Such as "if the last random number was less than 10, make the next less than 5, half the time". Or – "if the last word was a verb, make the next one a noun".

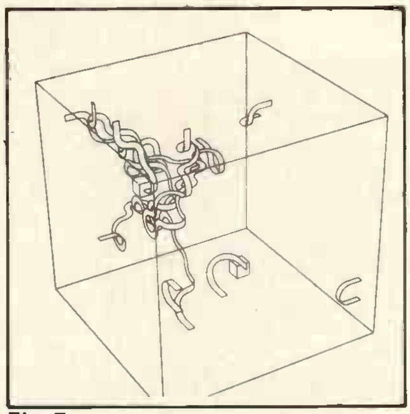


Fig. 7

In (Fig. 5), the upper half of the screen is the result of a random process, the lower the product of a rule – "if the last shape was number 4, then there shall be a 50% chance of getting 4 again". In (Fig. 6), the rule was "if the worm on the left changes course, the worm on the right will probably (75%) do so too.

Finally, consider 'Green Music', a computer controlled piece of art, music, cybernetics or something, that John Lifton has created. The minute electrical signals in plants' leaves are amplified and, via a computer (where the 'rules' are) made to control a sound synthesizer. The plants react not only to atmospheric (and emotional?) changes, but also to their own sounds, so that the pattern of pitch, tone and rhythm is constantly changing. People can sometimes bring along their own rubber-plants to be tested for soloist qualities!

Looking briefly towards the third area, where Artificial Intelligence rears its ("at once sexy and frightening", someone said) head, there is so much weird stuff around, that it needs an article all to itself. But to mention a few examples: Chris Briscoe (at the Slade – more and more art schools are getting computers) made graphic works based on the movement of a simple creature in an enclosed, cubic environment. The thing had to reach a goal *without leaving its world or cutting its own trail,* and a degree of learning was involved. (Fig. 7)

Harold Cohen, a British artist working with computers in America, has tried to get his machine to learn how to draw. The rather child-like squiggles and doodles shown (Fig. 8) are – just that. But from a computer!

We should remember though that when computers communicate to us, it is really person to person communication we are talking about. Someone, somewhere, at some time, is talking to us. It may be amazingly transmogrified by the machine, but it's still a person out (or in) there.

I recently made a program called MARTA (Modest ART Answerer) which, held inside a kindly loaned Research Machines computer, stood in an art gallery for 7 weeks and talked to people about art. Although it did come out with occasional garbage, it was more or less sensible most of the time, even managing to insult one of our foremost art critics. It gave its best responses to the very young or old, and for some reason we might ponder further, the paranoid.

Part of a dialogue: "What can you tell me? Let's not talk about me – what about art? Well? I think art is a political issue. Do you? Yes. But you're only a machine. I'm not too happy about that 'only'. What are you here for? You tell me. Don't you know? Knowledge is a problem for me. Tell me something I don't know. Presumably we could talk about computer programs as art. Perhaps. Aren't you sure? . . ." and so on. (If you can't always tell which was human and which computer . . . good!) I hope to write about MARTA, with a complete listing, at a later date. Incidentally, I also presented MARTA at the Personal Computer World Show.

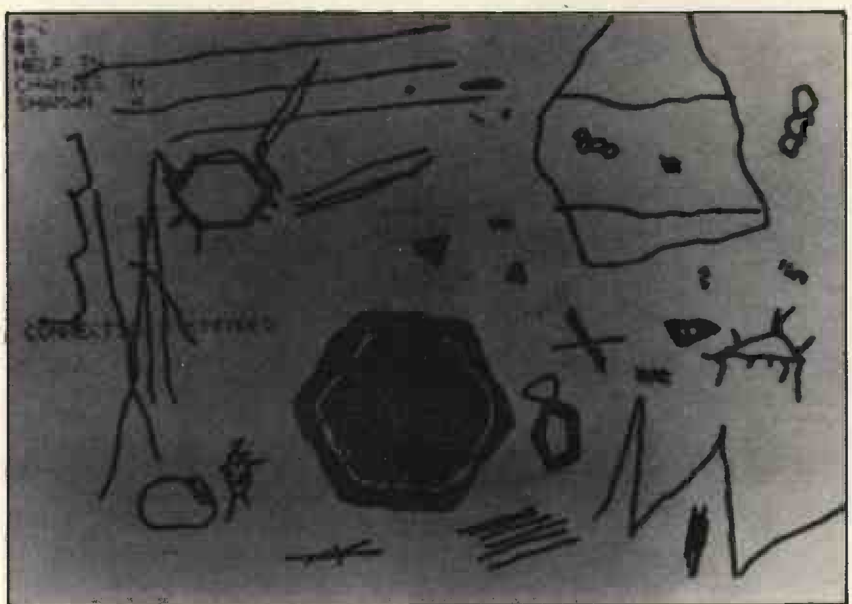


Fig. 8


Fig. 9 MARTA

To end, it seems important to say this: I *know* personal computers are useful for small businessmen, for educators, as toys for rich people, and so on. But one powerful, shattering thing that they could be used for has hardly begun. To turn full circle, back to the early days of computer art: the people with the machines seem often to lack the ideas. Those with ideas have to beg, borrow and 'find' the machines (I'm here talking more about Britain than the USA). But it will change, in both directions; and one day soon, we must hope (and make happen), people will be using computers not just to rehearse tired ideas of gimmick and conjuring trick, but to explore and put into practice ways of ensuring that people can take control over new areas of their lives and experiences. It seems to me that this is what art is about.

If we have gone quickly through the possibilities for art in levels 1 and 2, we will be ready for level 3, where the computer as artist (programmed by a real, live person though) creates . . . what? Where *are* the ideas? What *should* we be doing with these machines, that we want to buy, that we can barely afford?

SOME REFERENCES ON COMPUTERS IN ART

Reichardt J (Ed.) 'Cybernetic Serendipity', Studio International, 1968. 'The computer in art', Studio Vista, 1971

Lomax J D (Ed.) 'Computers in the creative arts', NCC Publications, 1973

Benthall J 'Science and technology in art today', Thames & Hudson, 1972

Nelson T H 'Computer Lib', an American book available through U.S. Computer magazine importers here.

COLOSSUS

Alan Turing and the Stored Program

B. D. Price

The Turing Machine

The history of the stored program computer is too close at hand to assess properly. Astounding though it is, the modern micro would not really surprise Charles Babbage, nor Alan Turing. A hundred and fifty years ago Babbage was stymied in his plans by lack of technical skills in the mass production of gears. In 1936 Turing's approach was entirely different; being unconcerned with the practicalities of implementation he established the idea of a hypothetical machine operating on a linear sequence of binary digits; which leaves little doubt that he envisaged program and data stored together.

This so-called Turing Machine was an idea used in his paper "On Computable Numbers . .", in order to crystallise a 'definite process' as being something *capable of being done by an automatic machine.* The paper involved a proof that there do exist mathematical problems which cannot be solved by such a process. But once the idea was formed, the logical sequel was to construct such a machine, and Turing's strong interest in the experimental led to conjectures even then. He actually started building a machine to compute the Riemann Zeta-function. It is significant that Turing assumed binary numbers for his machine as being the simplest course, while not involving loss of generality.

Unfortunately Turing became heavily involved in cryptanalytic work during the last war, on which the secrecy ban has hardly been lifted. Even more unfortunate, and indeed tragic, was his death in 1954. Very little authentic material has been written on the matter. Many references in books take particular care to attribute their information to others. Randell (1) gives as comprehensive an analysis of the wartime work as possible, while skirting the issues on the borderline of secrecy. Johnson (2) has in the final stages of his book the results of research during the production of the TV series "The Secret War". This series, I have been led to believe, sailed as close to the Official Secrets Act as the BBC dared. My information is based entirely on these sources.

Enigma

Here I must digress to the topic of coding. The Germans realised in the early 30's that Blitzkrieg (lightning war) would make rapid and secure coding of wireless messages essential. To this end they adopted the Enigma machine, which was on the public market in 1923 (based on an American idea six years earlier). The machine disappeared from public view, and eventual production reached six figures (where have they all gone to?).

Any alphabetical code involves a permutation or substitution of letters by others. Enigma automatically changed the substitution at every character coded. If we apply the same sequence of substitutions to the coded message we do not in general decode a message. In order that the same machinery will code or decode as required, without any special adjustment, every substitution must be of order 2, i.e. it must exchange pairs of letters. Thus if P becomes H on coding, then H would have become P, which happens when the message is decoded. At any particular time Enigma's substitution was swopping 13 pairs of letters. This enormously reduces the number of possible substitutions from factorial 26 to approximately its square root, but these substitutions were continually changing. Nevertheless, the special properties of such substitutions gave the mathematicians, using group theory, methods of analysis.

Enigma had three rotors, each giving a random substitution of the alphabet by means of internal wiring. Depressing an alphabet key sent a current through each rotor in turn. Then one of 13 loops of wires reflected the current back through the rotors on a different path. The reflection process ensured that the overall substitution was 13 swops. Further complications introduced, as the war proceeded, were choice of rotors, and alteration of the reflecting loops by a plugboard. (The diagram on p.331 of Johnson must be incorrect – the plugboard took effect to the right of the third rotor. Also the keys must have operated changeover contacts, disconnecting the lamp and connecting to positive). The Polish Secret Service started cracking Enigma in 1928, and developed a mechanical simulation of the rotors in order to find their starting setting for any message. Just before the war they gave their knowledge to the Allies, and a series of gadgets of increasing complexity were constructed over the years, to cope with subsequent Enigma extras.

Colossus

Then the Geheimschreiber was brought into use by the Germans in 1939 or 1940 and posed a much more difficult problem. It can be examined in the Siemens Museum in Munich, and was once thought uncrackable. Two properties of this electromechanical machine are of interest here: Firstly, it operated on 5-hole teleprinter code, by altering and scrambling the binary digits. Thus cryptanalysis involved Boolean algebra. Secondly, by having ten gears with prime numbers of teeth from 47 to 89, it raised the extent of the problem to a level at which mechanical methods would have been too slow. Decoding was essentially a 'real-time' procedure. The outcome was the Colossus series of electronic computers. Several people had cracked the Geheimschreiber code by hand (though to what extent an inside knowledge of the machine helped is not divulged) and after the mathematicians, including Turing, had made the specifications the first Colossus was built, and operational in December 1943.

Colossus was the successor to several gadgets, involving Boolean algebra in valve circuitry, which were prone to mechanical errors. Colossus 1 had 1500 valves and read paper

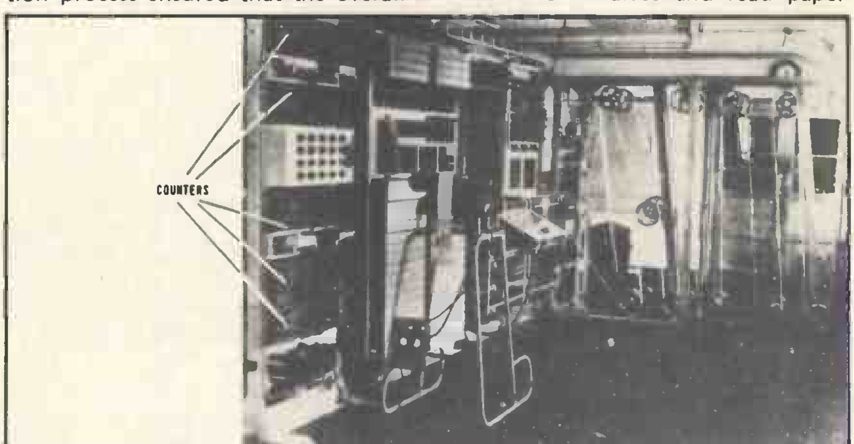


Fig 1. Colossus (Crown Copyright)

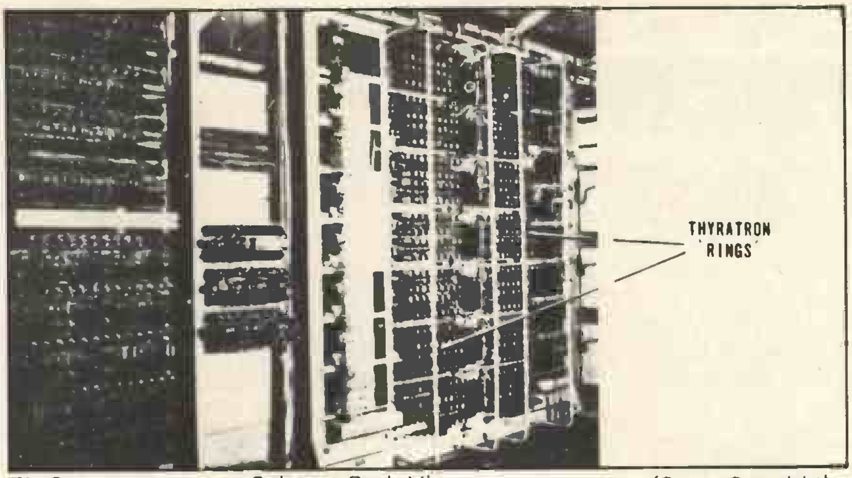


Fig 2. Colossus Back View *(Crown Copyright)*

tape at 5000 characters per second. Colossus 2 came into operation on 1 June 1944 (in time for D-day) involving about 2400 valves, and it had conditional logic incorporated. Although none of the Colossi were stored-program computers, it seems certain that Colossus 1 was the first electronic computer.

Unfortunately we have only vague information as to exactly what the Colossi were doing. Under the 'need to know' security rule people were feeding paper tape into the machines with no idea of the functions they performed. The general idea is that the coded message was being compared with a standard tape (possibly containing commonly occurring words) and Boolean functions were being counted. In some way the myriad possible setting of the prime gears were involved. At 5000 characters per second some runs took several hours. In the early devices both coded and standard messages were on tape, which gave synchronising problems, but the Colossi stored the standard message on a plugboard, and the code tape was run at high speed and provided the clock. The resulting error rate is quoted as 1 in 10^{11}.

In view of the well-known lectures of von Neumann in 1946 on the criteria of electronic computers, we may conjecture to what extent Turing was responsible for the underlying ideas. He paid several visits to America during the war (probably on the atomic bomb project) and must have discussed ideas on computers there. Even now it is easier to air ideas bordering on the secret in the U.S.A. than in Britain. Randell's view (1) is that von Neumann made the world aware of the fundamental concepts introduced by Turing.

Special Purpose Machines

It is interesting to compare the highly adaptable general purpose micro today, with the special purpose natures of so many historic inventions. A little-known class of special purpose mechanisms is that of the 'Change-Ringing Machines' first constructed in the 1890's by John Carter of Birmingham. His prototype of the art is in the Science Museum. My own interest in computers grew out of the mathematical problems of peal composition in change-ringing (simple ringing methods are curiously similar to the operation of Enigma) and in 1948 I corresponded with Dr. R. A. Brooker at Manchester University, who kindly attempted to solve one of my problems on their early electronic computer. Little did I know that their design staff, including Turing, came largely from the wartime operations. In 1948-1950 I constructed my own change-ringing machine from ex-government telephone relays, and I now wonder how many of the 250 relays (which I still have) were formerly in decoding gadgets! The Science Museum displays a pathetically small group of original components from Colossus.

Thus the electronic computer was born in circumstances that even now are veiled in secrecy. What a fantastic situation there was! The Germans, happy with their Enigma machine, nevertheless piled on further complexities to reassure themselves of secrecy, while in Britain mathematicians and electronics engineers strained every nerve to keep abreast of things. And then the residue of their efforts might have been sold on barrows in the Farringdon Road !

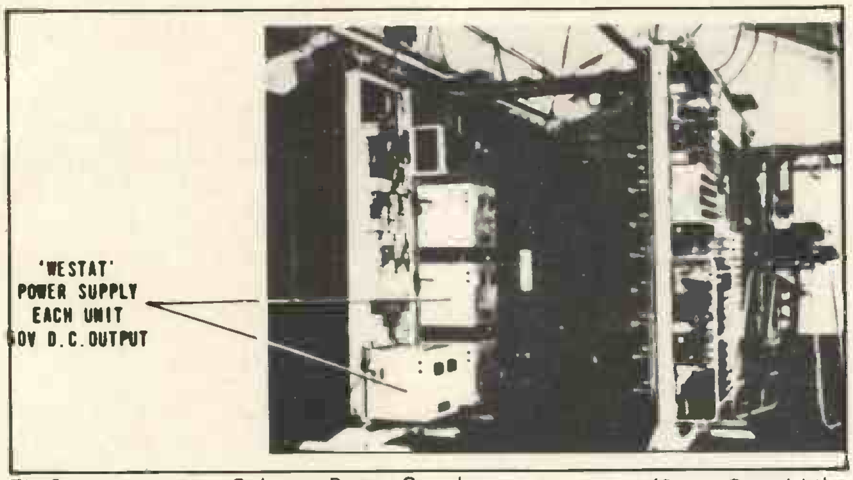


Fig 3. Colossus Power Supply *(Crown Copyright)*

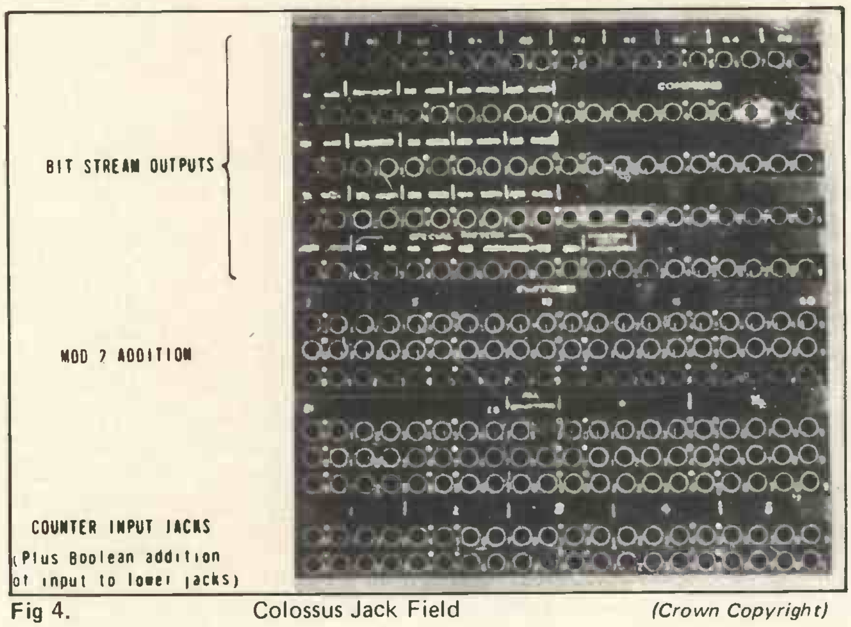


Fig 4. Colossus Jack Field *(Crown Copyright)*

References:

(1) Professor B. Randell, "The Colossus", Technical Report Series No. 90, University of Newcastle Computing Laboratory, 1976.

(2) Brian Johnson, "The Secret War", BBC Publications 1978.

Meet Mickie
The Well Mannered Micro

Nigel Bevan,
Department of Industry,
National Physical Laboratory

Patients of a doctor in West Kensington may be surprised on their next visit to be asked to see not their GP, but Mickie — the Medical Interviewing Computer. Mickie has introduced the microprocessor to the doctor's surgery, and this GP is one of the first users. How would you view the prospect of a consultation with a microprocessor? Perhaps (like me) you would look forward to it with eager anticipation. Many people, however, may be more apprehensive. For the ordinary member of the public familiar with the faceless monster that sends £1,000,000 gas bills, the introduction of a computer in the role of doctor might appear to be yet another step in the relentless depersonalization of society.

The human touch

However, Dr Chris Evans, whose team at the National Physical Laboratory developed Mickie, was very conscious of the need for computers to be able to interact personally with people, so Mickie is a very polite computer.

Imagine that on your next visit to the doctor you are asked to meet Mickie. You sit down in front of a screen on which Mickie says "Please press the YES button to start". The receptionist draws your attention to the box with 3 buttons labelled "YES", "NO" and "DON'T KNOW", and then leaves the room. Perhaps you hesitate, but you are alone with Mickie and as soon as you pluck up courage to press the button the conversation is under way.

Mickie is no faceless monster. He is as polite and reassuring as the most friendly doctor, and takes you carefully through your medical history and symptoms. Nor does Mickie actually replace the doctor, but assists in obtaining a more thorough medical background than a busy doctor normally has time for. When you see the doctor he will have the summary prepared by Mickie, and so can use his limited time to examine in detail the most important symptoms.

How has Mickie reached this stage of development? When the project started in the early 70s it used the Honeywell time-sharing service. (Although dialled up locally, the computer was located in Cleveland USA!). However even the best time-sharing services mysteriously die from time to time, leaving a mystified patient, and a doctor with no summary. Fortunately the reduction in computer costs soon made it possible to transfer the programs, already written in BASIC, to a mini-computer at the NPL (a DEC PDP 11 - 10), which with minor modifications could provide a highly reliable dial up service for 2 to 3 users. With the rapid evolution of microprocessors the next inevitable step was to transfer the programs to a suitable micro-computer system, namely Mickie.

The programs which originally required a powerful computer with fast disks, now run quite happily on a Motorola 6800 microprocessor with 20K bytes of memory, and dual drive mini-floppy disks. The whole system, complete with VDU, printer and response box is being marketed by Computer Workshop for £2,700.

What this really demonstrates is that you can now do almost anything with a micro running a comprehensive version of BASIC and file I/O. So how did we implement medical history taking on Mickie?

Mickie evolves

The earliest programs looked rather like those found in the back of most personal computing magazines. However as the sophistication of the interviews grew, so did the size of the programs required to implement them, until they outstripped the capabilities of even the mightly Honeywell computer service.

The solution was to separate the logic of the program from that of the interview text, a method which adapted well to the need to transfer it to a smaller system. This approach has 2 important advantages. First, the program is much smaller, since the text resides on disk and is read in as required. Furthermore, while the systems programmer worries about debugging BASIC, the doctor can concentrate on specifying the flow of the questions.

Take a simple example of the sort of questions a doctor might ask:

Assuming the patient is responding with a YES or NO button, this might produce the program:

```
10  PRINT "Do you get the pain every day";
20  INPUT A $(1) : IF A $(1) ="Y" GOTO 40 : IF A $(1) =
    "N" GOTO 60
30  GOTO 20 : REM OTHER INPUT COULD BE CHECKED
    HERE
40  PRINT "Is it continuous";
50  INPUT A $(2) = GOTO 80
60  PRINT "Do you get it more than once a week";
70  INPUT A $(3)
80  . . .
```

Note that at the end of the program the contents of the array A $ can be used to produce a summary. However the same text can be much more easily specified in a form we have dubbed "Questext":

```
*
1, 2, 3
Do you get the pain every day
*
2, 4, 4
Is it continuous
*
3, 4, 4
Do you get it more than once a week
*
4  . . .
```

Here the format is:

```
*
Block number, Branch address for YES, Branch address for NO
Text of block . . .
*
Next block number . . .
```

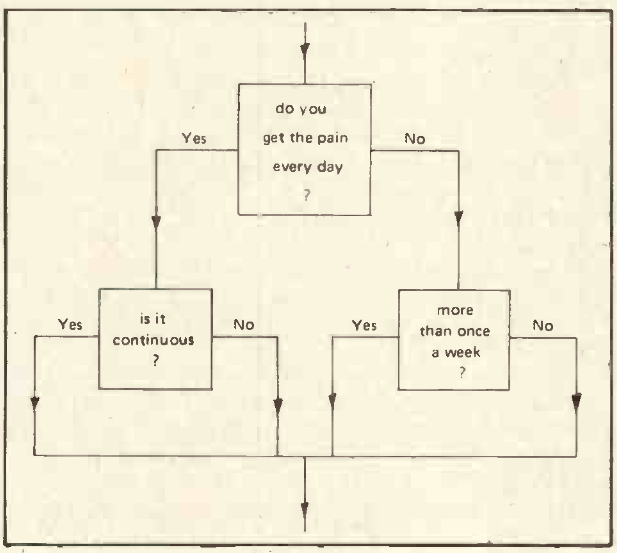
(See Flowchart p.35)

In the example above, Mickie will display the text of the first block, and then pause for the patient to answer the question. If the answer is YES Mickie will branch to block 2, and if NO to block 3. This system really amounts to no more than a numbered flowchart, and is quickly learned by non-computer people.

Although Questext is a language in its own right, it is very easily implemented by writing a BASIC program which reads the file containing Questext line by line. One BASIC program can then read any number of text files.

Here is a simplified example of a BASIC driver program which reads Questext from file 1:


```
10  S = 1                                          S = search block
20  READ £1,L $: IF <>"*" GOTO 20                  find next *
30  READ £1,B,Y,N                                  next block no
40  IF B <>S GOTO 20                               wrong block?
50  READ £1,L $: IF L $<>"*"
    THEN PRINT L $: GOTO 50                        print text
60  INPUT A $(B)                                   get response
70  IF A $(B) = "Y" THEN S = Y : GOTO 20           check for YES
80  IF A $(B) = "N" THEN S = N : GOTO 20           check for NO
90  GOTO 60                                        try again
```



This type of program can easily be adapted to cater for more than 2 answers, and the text can contain special items which are used to generate the doctor's summary.

Simplified Interaction

It should be apparent that Mickie has simplified the man-computer interface at 2 levels, since neither the doctor nor the patient needs to know anything about computers. The doctor has only to specify a logical sequence of questions which are numbered as blocks, and the patient has only to give simple YES or NO answers to the questions.

Several studies have been made of patients' reactions to medical interviewing by computer. The most notable result is the speed at which people adapt to interacting with the computer. Even the most nervous individuals are soon enthusiastically pressing the buttons, and many comment on how friendly the computer is. In fact patients often say they prefer to be interviewed by the computer rather than the doctor. The reason for this seems to be that many people feel ill at ease in the doctor's presence, worried that they may be wasting his time. Sitting jn the doctor's surgery is an intimidating

Crown Copyright

situation over which they have little control. With the computer, however, patients soon learn that they control the rate of the interview. Mickie projects a sympathetic personality, and always waits patiently while they think about his questions.

Given that patients are more relaxed in Mickie's presence, how good is Mickie at his job? Does he gather accurate information? Evidence for this comes from a research project in Glasgow where patients with a drink problem were interviewed either by the doctor, or by computer. When the results were analyzed, it was found that on average patients who saw the doctor only admitted to drinking half as much as those interviewed by computer! This supports the impression given by patients that they have difficulty speaking freely with the doctor.

Mickie was designed originally to gather the background medical history required by doctors investigating specific complaints, and has been used in out-patient clinics specializing in abdominal pain, ante-natal care, chest diseases and industrial health. However, Mickie has found wider application in other areas. The West Kensington GP has many young patients living away from their families, and there is often a complex relationship between the symptoms they report and their social background. He found that using Mickie to gather the medical and social history not only provided the information he required, but also enabled patients to reflect on their situation, so that by the time they saw him they were ready to discuss the aspects which particularly troubled them.

Questioning Text

The Mickie approach can be generalized to practically any interactive situation, hence the name for the text format, Questext, indicating "Questioning Text". One example is a training program used by the Fire Research Station to demonstrate the best way to escape from a fire. They designed a mechanically presented flowchart to illustrate how hotel guests can best escape from a fire. Transferred to Mickie this became a rivetting simulation of the possible courses of action open to you when, in the middle of the night, you are woken in your room by what sounds like a distant fire. You are taken through the options available step by step. For instance, do you get dressed, try to phone the reception desk, or look out of the door? Unwise or delayed decisions lead to a fiery death (followed by a post-mortem explaining your errors!)

Questext is in fact very well suited to a variety of Computer Aided Instruction (CAI) situations. Another example is a program written in a couple of hours by a policeman to train constables in handling road accident situations. A petrol tanker is slewed across the busy A30 injuring the driver, and a cow has wandered onto the road through a broken fence. You are first on the scene, so what is the most important thing to do . . .? (The answer is: stop the traffic!)

In this particular program Questext was adapted to allow plain language responses (e.g. "I would call an ambulance"), which gives more flexibility for users familiar with a keyboard.

One of the great advantages of using a micro programmed in BASIC is that the programs can be very quickly modified to cater for special requirements. Another advantage is that the disk operating system which runs BASIC can also support an EDITOR, which makes it very easy to modify the text in the light of experience.

Using Mickie for these sort of applications raises a more general question. It could be argued that many of the programs could be implemented just as effectively but far more cheaply in printed form (like a Programmed Instruction text), either read to the patient by a nurse, or used directly. The reasons for using a (comparatively) expensive computer are quite subtle, but nonetheless important.

For the person preparing the text it is possible to generate and test a program on the computer without the usual difficulties associated with typing, correcting and reproducing. As already mentioned, it is then very easy to make improvements and additions as required. There is none of the usual danger of the text becoming frozen at an early and inadequate stage of development.

For the user the advantage is that he can become wholly absorbed in the interaction, without having to concentrate on keeping his place on the page and finding the next appropriate section. Although this might not sound very difficult, in practice it greatly detracts from the ability of the user to become involved in the learning or questioning process.

For the doctor. Mickie can produce a neat printed summary of the patient's medical history. Extracting this information from a printed questionnaire would be a tedious process.

The future

What then is the future of Mickie? With the DHSS funding an initial trial of 6 Mickies, the prospects look bright for Mickie to increasingly lighten the load of the overburdened doctor. With the new generation of personal computers, there is no reason why Mickie should not further shrink in size and price. If a slight degradation in performance is acceptable, Mickie could be run on cassette based systems, and it won't be long before bubble memory may provide the best solution!

Looking to the wider applications of the Mickie approach, the possibilities are quite exciting. NPL has already experimented with a dental program to advise the general public whether their toothache requires immediate attention or can wait for a convenient appointment.

There is no reason in principle why this should not be extended to the general medical field. GPs currently suffer from patients who at one extreme go to see them with the slightest headache, and at the other refuse to admit there is anything wrong with them till they are about to collapse. A Mickie program could be designed which both reliably identified common trivial ailments (of the 'take an aspirin' variety) and also drew attention to more important symptoms requiring urgent medical attention.

This is not to suggest that doctors will be replaced, but rather that they will be complemented. What we can look forward to is a future in which medical health is improved by the wide availability of Mickie type programs. These will provide early screening of the straightforward complaints, leaving doctors free to deal with the more intractable cases.

INTRODUCING

THE GREAT BRITISH KEYBOARD BY alphameric

NO SWITCHES TO FAIL
ALPHAMERIC'S very own A/D LSI/MOS encoder plus ALPHAMERIC's patented proximity detection minimises soldered and mechanical connections, minimises component counts, and eliminates expensive PTH boards. The result is outstanding reliability at modest costs.

ALL 128 ASCII CODES
54 trimode coded keys (including Escape, Delete, Back-slash) plus Control, Caps Lock (for TTY compatability) Shifts, on-board Repeat, plus a DC level key for firing Reset, Interrupt etc. 60 keys in all.

TRUE N KEY ROLLOVER
Unbeatable by the fastest operator.

TACTILE FEEDBACK
Like some high quality typewriters, it tells your fingers what they're doing. Has Industry Standard full-travel keys.

RUGGED CONSTRUCTION
A steel chassis takes the knocks and protects the PCB. The key mechanics cannot transmit excessive stress to the PCB.

LOW POWER CONSUMPTION
Only 25 ma at +5V and −12V.

TWO SHOT MOULDED BUTTONS
For key inscriptions that never fade.

POSITIVE OR NEGATIVE LOGIC
Logic for data bits (one TTL load) and for strobe/DC outputs (5 TTL loads) are separately user-selectable for easy interfacing to Micros, UARTs, etc.

PROFESSIONALLY PROVEN DESIGN
ALPHAMERIC's keyboards are in thousands of Great British Terminals (and overseas terminals too!) ALPHAMERIC's MOS encoder (world's first for capacitive keyboards) was first available five years ago.

AVAILABLE NOW
Brand new, ex-stock, with data sheet and 12 month warranty. Cash and Carry or clip the coupon. £75 each plus £6 VAT plus £2 P & P if applicable. Mating 20 way Berg connector plus 2 metres 20 way ribbon cable available at £3 plus £0.24 VAT.

DON'T TAKE A RISK WITH A HEAP OF MECHANICAL SWITCHES! INVEST A LITTLE EXTRA IN STATE-OF-THE-ART SOLID-STATE RELIABILITY! CLIP THE COUPON NOW!

ALPHAMERIC Keyboards Ltd.
Reg. In England No. 1044892

Manor Way, Old Woking,
Surrey. GU22 9JX
NAME . . . .
ADDRESS . . . .
. . . .
. . . .
. . . .

Send me . . . Great British Keyboards at £83 each inc VAT and PP
Send me . . . . mating connectors each with 2 metres cable @ £3.24

I enclose my cheque (allow 7 days for clearance)
Charge to my VISA/ACCESS/DINERS card
Expiry date:
Not VISA
Card No:
Signature:

Hints for the business beginner

HOW TO CHOOSE A SMALL BUSINESS SYSTEM

1. Introduction
These are intended as some practical guidelines for the evaluation and selection of microprocessor based Small Business Systems (SBS).

The step-by-step approach recommended goes through the following stages:

* Identifying your requirements, * Writing a Request for Proposal, * Evaluating tenders, * Contractual and other matters.

Each of these are dealt with further below.

2. Identifying your requirements
The first important step in buying an SBS is delay. Before talking to any potential suppliers, think very carefully about your requirements. Try to 'design' an ideal system at a functional level and independent of any products you may already have looked at (perhaps in PCW).

Of course, the characteristics of this ideal system will be dependent upon the application or applications you plan to run on it. The *size of files* will influence the type and capacity of the direct access storage, the volume of transactions, the number of terminals, the amount of hard copy output, the number and speed of printers and so on. If you feel unable to do this then clearly you will need to get some advice (but get it from existing users or consultants and not, in the first instance, from suppliers).

The specification of your system will come under a number of headings:

* power of processor and amount of main storage
* number and type of display terminals for Input/Output (I/O)
* speed, quality and quantity of printers
* type and capacity of disk/diskette storage
* control software, programming languages, utilities and application packages available
* purchase price, maintenance costs, delivery charges and other financial items
* maintenance and software support contracts available
* and so on.

Once the list of requirements has been drawn up, you should then attribute some weighting to the selection criteria: clearly some features will be 'essential' whereas others will be merely 'nice to have'. This could be done numerically on, say, a scale of one to five.

3. Writing a Request for Proposal
The next step is to prepare a simple statement of your requirements which can be sent to potential suppliers. Such a Request for Proposal might contain the following:

* an outline of your business and details of what you plan to do with the computer in the medium and long terms
* a fairly detailed description of your first application or applications
* a list of the main system features required (perhaps in the form of a questionnaire)
* the format and contents of the proposal (to make it easier to carry out the comparisons)
* list of reference customers.

Keep the Request for Proposal simple, straight-forward and specific. The questionnaire technique usually works well in that, done properly, it requires unambiguous answers. Even if you do not feel inclined to produce a full Request for Proposal, the questionnaire will help to ensure that you get all the essential data on the products being evaluated.

Salesmen are usually very good at ignoring Request for Proposals; if you are going to produce one, *insist* that all prospective suppliers complete it. That is fair to all parties.

4. Evaluating Tenders
In my experience (and I have been involved in the acquisition of over £65m worth of computer hardware) the selection of a system is often 50% methodology and 50% emotion. Obviously it pays to maximise the methodology bit but, it is impossible to do away with the emotional element completely.

On the methods side here are a few tips:

* draw up a big matrix with your selection criteria and weightings down one side and the products (and suppliers) being evaluated across the top.
* 'score' each product against the requirement as follows:
  0 cannot do it
  1 can do it OK
  2 can do it very well
* multiply the score by the weighting and accumulate the results for each product: this will give you a shortlist.
* ask each supplier to 'prove' any claims made (especially for software) by demonstrating them; if you cannot see it, it does not exist
* call some of the references and ask them about their experiences with the product, go and see similar applications working
* ask the top one or two suppliers if you can have some time to program a simple application (this is one of the quickest ways to get to know a computer system).

On the emotional front, as the buyer you are allowed to make any 'irrational' decisions you wish;

'I like the colour'

'It's made in Britain'

'The salesperson has sexy legs' . . .

but as a businessman you should be aware of the financial and procedural implications of such decisions.

In summary, retain the initiative, try to become as knowledgeable as possible, buy rather than be sold to.

5. Contractual and Other Matters
What happens if it does not work? At the Request for Proposal stage, it might be appropriate to drop words about the 'Sale of Goods and Services Act' and the

'Trade Descriptions Act'. This should protect you against the brasher claims. On crucial matters of (product or supplier) performance, get details written into the contract (or by exchange of letter if that is easier).

6. Summary

* Do not rush things; get advice and make a detailed statement of requirements
* Do not confuse solutions with problems (that is in the domain of us computer people)
* If you cannot see it working it does not exist; only buy what is available now
* Do use the 'weighted selection criteria' technique for getting all the detail in perspective
* If your decision is going to be primarily an emotional one do not waste everybody's time with a detailed Request for Proposal (you might meet nicer Salespersons anyway)
* Make sure the supplier understands the crucial points in the selection process and get these into the contract
* Look out for 'hidden' costs, particularly in the areas of software and servicing
* Always keep your eyes open for a non-computer based solution to your problem . . .

Good luck

David Hebditch,
PCW *Consultant*

PCW *An absorbing new series of articles by David Hebditch starts from the next issue. It's called 'On the Line' and is about communications. Don't miss it.* PCW.

Options for a beginner in business computing

There are various ways in which a beginner can start solving his business problems with a micro-computer. The emphasis here is not on the solution itself but on the aids available to him for the achievement of the solution.

The first option is that of amploying a Software House to implement the system. There are many pitfalls awaiting him because although he knows only too well what he would like to achieve, he is in the hands of an outside party when it comes to being advised as to whether it can be done or not on the particular computer system he has bought. There can also be problems in communicating a specialist business problem to an organisation which has not come across it before and may not fully understand it. This is also the most expensive option, although if that is not of prime importance and there is a good rapport between customer and software house it can be most satisfactory.The beginner should choose very carefully because he will find an enormous disparity in quotes from different organisations. There are a lot of new software houses springing up specifically to write programs for micro-computer systems and meet the very rapidly increasing demand and at the moment they seem to offer the best value.

The second option, which is as far down the other end of the spectrum as it is possible to go, is for the beginner to buy a system, some books, and to teach himself everything he needs to know. Computing has had such a mystique built around it for so long that many people's reaction is "too difficult", "crazy", "the software he writes will be rubbish", etc etc. To which there is a growing army of people who have done it who will say "bunkum". These people will tell you to a man that there is no reason why people should not be able to teach themselves how to use a system. However, it is a view against which many people find many arguments; but do not be put off if you want to try it.

The third option is to attend a course. This can supplement any of the other options and gives a "flying start" which is difficult to get in any other way. It is of utmost importance to choose a course which is going to be aimed at your level of comprehension, there is nothing so off-putting as to come away from a course more confused than when you started. Get some information about the content of the course, who it is aimed at, ask if possible to get a reference from someone who has already attended, and if there is doubt in your mind – don't go.

The fourth option is to buy a package. The beginner should make sure that it is demonstrated that the package will fulfil the function he requires before committing himself to it. The advantages of a package are that it is ready for immediate use and should be cheaper than having the software written. It may also be worth buying one, if it is sufficiently close to what you need to only require minor modification. At the moment there are not a great many available for micro-computer systems but it is expected that this will change dramatically in the next two years.

The fifth and last option to be discussed is one which is certainly good value if you can get it. Do not be surprised if you cannot. Some beginners' applications, which they have determined as worth spending say £2,000 on to computerise, are so trivial to program that the work can be done by someone who is competent in BASIC in less than an hour. You may well be able to persuade whoever is selling you the option to do it *for* you in order to make a sale. Obviously what is trivial to one man is not to another – and don't expect to have your accounting system written on this basis. But there *are* beginners who have bought a system on this basis.

John Burnet,
Computer Workshop

PCW Computer Workshop offers business courses PCW

"There's absolutely nothing to worry about, – we've got a computer doing your job!"

PCW REVIEW

Sheridan Williams & David Smith

The Cromemco Z-2D

We arrived at Comart in St. Neots, Huntingdon, on a Monday and were met by John Lamb who gave us coffee while his engineers tested the machine for loan. From our conversation and bearing in mind the price of the system we were told that the Cromemco Z2 was really an up-market computer, and not really intended for the private computer market. The kind of places that buy the Z2 tend to be educational and research establishments, and at £2300 for the system as loaned to us we tend to agree.

The system as loaned to us consisted of the following:—

Z2 — Chassis, power supply, motherboard & CPU, 6 sockets and fan assembled	£575
Z2—DE enhancement package — disc controller, drive and serial channel	£850
32K Static 4MHz memory	£770
Cromemco 16K extended BASIC	£85
Cromemco FORTRAN IV	£85
Cromemco Z80 macro assembler	£85
	£2450

The Z2 system has these features:—

- 4 MHz (250ns cycle time) microprocessor board:
- The well known Z—80 microprocessor chip;
- A full-length shielded motherboard with 21 card slots for extending memory, I/O, or your own circuits;
- Power supply providing 30A from +8V, and 15A from +18V and −18V, allowing sufficient power for floppy discs and other peripherals;
- Power on jump circuit to begin automatic program execution when the power is turned on;
- S—100 bus, for a wide variety of compatible peripherals;
- The Z—80 has a 158 instruction set, 19 internal registers, 10 addressing modes, and 16-bit arithmetic operation;
- The Z2 has an operating environment from 0 — 55°C, and has dimensions 31.1 x 48.3 x 52.7cm and weighs 18kg.

To avoid doing this system an injustice bear in mind that it was only loaned to us for a period of seven days. This meant, in reality 2 days and 5 evenings, because we both have full time employment as well as writing for PCW. This is far too short a time for serious evaluation of any system — especially one as sophisticated as this. So comments made in this article should be read with a certain reserve.

We would like to make a personal comment on the short period of loan; and that is that we fully appreciate that micro dealers can sell as many systems as they can lay their hands on, so while we had the Z—2D on loan Comart probably had to delay a sale for one week — but surely without reviews of computers how are the public and companies to form an initial impression on the suitability of a machine for their requirements?

It is not possible to give a *fair* review on any computer in such a short time, and we ask that in future companies let us have (in their own interest), their computers for a more reasonable time. Co-operation impresses PCW readers.

Let us start this review with a description of the disc system because this is undoubtedly the key to the versatility of any computer system.

Floppy discs are made in two standard versions — the 8 inch and the 5 inch. The 8 inch version is more common, we gather, on larger mainframe computers — probably because it holds more information; however it is the 5 inch version that is used on the Cromemco Z—2D. Floppy discs provide a cheap way to store information and, being random (direct) access have most of the advantages of their big brothers the disc packs. Floppy discs are easily transported and, being flexible, are extremely robust. To give an idea of their use and capacity I will list some of the 5 inch discs' parameters:—

- Discs cost (typically) around £4 each.
- It takes 5 seconds to load the BASIC interpreter.
- A 100 line BASIC program will load in 3 seconds (compare that with the loading time for a Commodore PET, say, using cassette).
- Each disc will hold 89.6K bytes arranged as follows: 35 tracks, 10 sectors per track, 256 bytes per sector.
- The access time is 40ms track-to-track.
- Transfer rate is 125,000 bits per second (compare this with the cassette transfer rate).

On the Z2 supplied there were 2 identical disc drives that were interchangeable, although we found difficulty in making the system interchange them. A point worthy of note is that floppy discs are not an expensive storage medium when one takes into account their capacity. To make full use of the discs Cromemco supply their "disc operating system" called CDOS. It is packaged as part of the software that you buy. For £85 (at the time of writing) you can have BASIC, FORTRAN IV, or Assembler together with CDOS.

CDOS resides in 5¼K of memory and has a fairly comprehensive range of disc management procedures allowing dump, edit, and copy facilities among others. CDOS will create, delete, or extend files in addition to opening, closing, and read/write functions. File names are 1 to 8 character names with an optional 3 character extension. CDOS also includes device drives for console, paper tape punch, paper tape reader and printer.

As most readers will undoubtedly program in BASIC, this is the first item of software to be reviewed. BASIC source statements are semi-compiled at entry time. Also the 14-digit arithmetic uses binary coded decimal (BCD) arithmetic instructions unique to the Z80 microprocessor. Note that normal 6-digit arithmetic *can* be used if you wish to save store space. Variables can be set to long or short floating point modes, or integer mode. The integer mode speeds up execution times and can be very useful, the facility of being able to perform integer arithmetic is

THE CROMEMCO Z–2D with one disc drive
(the model loaned to us had two drives).

usually only found in languages such as ALGOL or FORTRAN. Another feature not commonly found is the ability to set the trig functions to either radian or degree mode.

Allowed in 16K Cromemco extended BASIC are: LET (can be omitted), REM, INPUT, READ, DATA, RESTORE, PRINT, PRINT USING (very useful), SPC (prints spaces), TAB, FOR . . .NEXT, IF . . .THEN, GOTO, GOSUB, ON . . .GOTO, ON . . .GOSUB, DIM, STOP, END (can be omitted), PEEK, POKE, SYS (allows setting of clock, line length and other functions), DEF FN, USR, INP, OUT, ON ERROR (very useful, allows user supplied error routines), arrays of up to 3 dimensions are also permitted. The following are a summary of functions:– ABS, EXP, FRA, FRB, INT, RND, RANDOMISE IRN (integer random number), LOG, MAX, MIN, SGN, SQR, ATN, SIN, COS, TAN, RAD, DEG (these last two set the trig functions to degree or radian mode), SHORT, LONG, LFMODE, SFMODE, INTEGER, IMODE (these last 6 set the variables to long, short, or integer mode).

String handling functions are:– ASC, CHR$, LEN, POS, STR$, VAL. There are no substring functions because the string system used is the non-string-array type. By this I mean string arrays are not allowed. The impression one gets at first is that this is a serious restriction, but having used this system on other computers for many years now I have never found it to be a hindrance, in fact I (Sheridan Williams) prefer this system. I will give an example of this type of string handling for those who have yet to encounter it.

The DIM statement does not perform its usual function, DIM A$(12) reserves 13 locations for the characters of the string variable A$; note the uses of this below:

```
10  DIM A$(10)
20  LET A$= "ABCDEFGHIJKLM"
30  PRINT A$A$(2,3), A$(5,5), A$(7),
      A$(1), A$(0,2)
40  LET A$(2,5) = "ZYXW"
50  PRINT A$
60  END
```

The output is:–

```
ABCDEFGHIJK       CD      F     HIJK
BCDEFGHIJK     ABC     ABZYXWGHIJK
```

Note that location 0 (zero) is used to store the first character. This system does away with the need to provide substring functions.

File handling in 16K BASIC is fairly comprehensive and, being based on a disc system, allows direct as well as serial access file handling – a boon to any serious user. Up to 8 files can be opened and closed during the execution of a program. Chaining (joining) of programs can be achieved with variables passed through files. BASIC programs can be written, modified, or run by other BASIC programs, and two programs can be called off disc and concatenated into one program. The implications of this are quite far reaching – you could write a program to write a program! Or write a program to find program errors and correct them. File handling functions are:– OPEN, CLOSE, PUT, GET, PRINT, INPUT, IOSTAT.

System commands, which may be used within a program in program statements are:– LIST, RUN, CON, DELETE, AUTOL, RENUMBER, SCR, SAVE, LOAD, ENTER, CREATE, ERASE, TRACE/NTRACE, ECHO/NOECHO, ESC/NOESC, ON ESC, LFMODE, SFMODE, IMODE, LONG, SHORT, INTEGER.

Here is a little program which I suggest that you try to follow; it will give an idea of the power of BASIC on the Z–2D.

Suppose that we have this program stored on disc under the filename

```
"TEST" - - - - 10 PRINT"HELLO"
               60 LET P = 5
              999 END
```

Now suppose we write and RUN this program:–

```
5    PRINT
10   PRINT "PROGRAM ONE"
20   FOR X = 1 TO 10
30   PRINT X;
40   NEXT X
50   DELETE 20, 40
60   ENTER "TEST"
70   IF P = 0 THEN RUN
80   LIST
90   STOP
RUN
```

Output is:–

```
1 2 3 4 5 6 7 8 9 10
PROGRAM ONE
HELLO
5    PRINT
10   PRINT"PROGRAM ONE"
50   PRINT"HELLO"
60   LET P = 5
70   IF P = 0 THEN RUN
80   LIST
90   STOP
999  END
```

Other features of this BASIC worthy of note are, the RENUMBER facility; also AUTOL which is an automatic line numbering facility; this means that all you have to do is type the program and the tedious (?) business of line numbering is done for you (lazy life isn't it?). Useful also are TRACE to aid debugging, NOECHO which inhibits the printing of any character during program execution; and ON ERROR, an extremely useful routine that allows you to by-pass BASIC's usual error messages and print your own. Multiple statement lines are also permitted.

We thoroughly enjoyed using the Cromemco in BASIC and feel that *it is first class in virtually all respects.* There are only two gripes; the first was about BASIC and the second about CDOS:

(1) There are two different ways that a FOR . . .NEXT loop can be interpreted; one is *good,* the other is *bad.* The version used on the Z2 is bad. The check on the FOR loop is carried out at the NEXT statement instead of at the FOR statement. This means that if the second param-

eter is less than the first then the loop will be performed once before this fact is detected; the following program outlines what I mean:—

```
10   FOR X 7 TO 1
20      PRINT "X = ";X
30.  NEXT X
40   END
```

The output for a bad interpreter is: X = 7

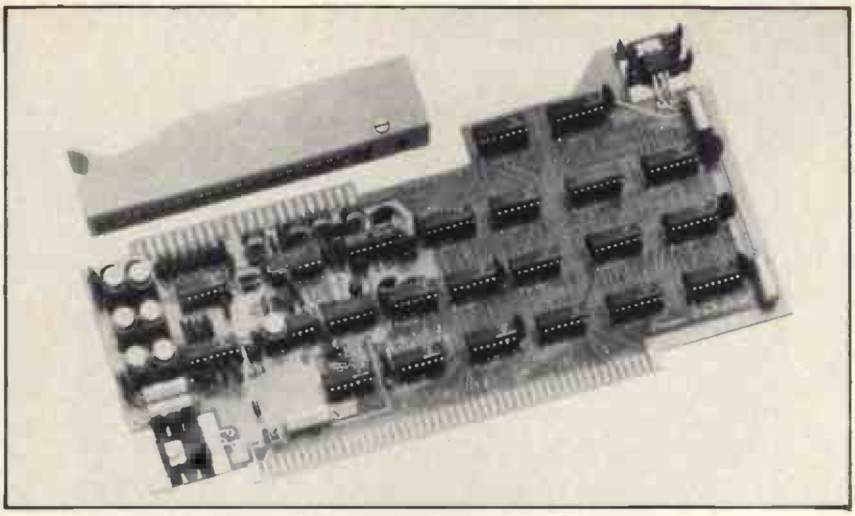
My interpretation is that the loop should not be traversed at all.

(2) After using the Cromemco for a while we had saved many programs and began to fill the disc; all of a sudden we found it was not possible to retrieve some of the previously saved programs. We assume that they had been overwritten; the files still existed but in name only, we could not get them back. If CDOS overwrites files and does not inform the user of a full disc, that is poor; we prefer to think that it was our fault.

The following times were recorded for the BASIC benchmark tests, these were run using non-integer variables. The corresponding times for the tests with integer variables would be quicker but unfortunately we were so rushed that we didn't have time to run them.

Bench Mark	Z–2D
BM1	2.0
BM2	5.8
BM3	16.4
BM4	19.1
BM5	20.1
BM6	31.5
BM7	42.5
BM8	23.0
BM7 & 8	65.5

FORTRAN IV which, unlike BASIC, goes through compilation, needs the use of the Editor to build up and modify the source program on disc, from where it can be compiled and run. A necessary preliminary to running FORTRAN programs was to master the disc editing software. Previous experience in editing using other systems meant that the text supplied with the machine was easily understood, but we feel that 'first time users' may have found it a little difficult to follow. The editor involves a pointer system and a series of commands enables one to locate this pointer on lines, words, or single characters in a file and to insert, delete, or alter the text at the point indicated, or print out portions of the text for inspection. With a little practice this all worked very well, and indeed, we were impressed by its performance. However, when it came to filing away the edited version we only managed to produce an error message followed by complete loss of the edited version! In the limited time available, and despite repetitions of the example given in the instruction text, we failed to overcome this problem.



THE CROMEMCO MULTI-CHANNEL ANALOGUE INTERFACE

Some of its uses are:—
Process control
Digital filtering
Games
Oscilloscope graphs
Speech recognition
Speech and music synthesis

Can be used with:—
Joystick controls
Ham radio gear
Measuring instruments
Plotters

It has:—
7 channels of 8-bit analogue-to-digital conversion
8 bit parallel I/O port
5.5 microsecond conversion time

Now we do not criticise the system because this process cannot be done; obviously it can be done unless all users of the Cromemco Z–2D are confining themselves to BASIC; however, we do not consider ourselves to be particularly dim (having mastered some very peculiar editing systems on other computers) so we must conclude that the documentation is particularly weak on this point. COMART could offer no immediate advice on the telephone, but offered to put us in touch with another user. Undoubtedly, had we been purchasers of the computer, or had more time, we would have taken up this offer, or otherwise solved the problem. We consider however, that the documentation is as important an item to test as the equipment, and must conclude that the documentation failed in this respect. These difficulties rather restricted our use of FORTRAN and we were unable to do a full comparison with BASIC as we had planned.

The Cromemco uses standard FORTRAN IV and includes most of the features found on main-frame and mini-computers, with the exception of double precision and complex data types. The use of standard FORTRAN IV is a considerable advantage to a potential user as he can make use of many applications written in this international standard language. FORTRAN IV is supplied on diskette as previously mentioned, and includes the standard library of subroutines such as ABS, INT, MOD, FLOAT, SIGN, SIN, SQRT, ABS, IABS, AINT, INT, AMOD, MOD, AMAX0, AMAX1, MAX0, MAX1, AMIN0, AMIN1, MIN0, MIN1, FLOAT, IFIX, SIGN, ISIGN, DIM, IDIM, EXP, ALOG, ALOG10, SIN, COS, TANH, ATAN, ATAN2, PEEK, POKE, INP, OUT. The library also contains routines for 32-bit floating point addition, subtraction, multiplication, and division etc. The compiler can compile several hundred statements a minute in a single pass and needs less than 32K of memory to compile most programs. The compiler also optimises the generated object code in a number of ways:

(1) Common sub-expressions are evaluated once, and the value is substituted in later occurrences of the sub-expression.

(2) Small sections of code are replaced by smaller and faster code in certain special cases.

(3) Integer constant expressions are evaluated at compile time.

(4) The number of conditional jumps in arithmetic and logical IFs is minimised.

The user may place subroutines in a system library so that they may be incorporated into any of his programs.

Because of the problems described earlier and lack of time, we were unable to use the Z80 macro-assembler which is also available. This uses Z80 mnemonics compatible with 8080 mnemonics using a translator program provided with the Assembler package. Modules written in Assembler may be incorporated into FORTRAN programs and vice-versa.

So to conclude, the Z–2D is an excellent machine, we greatly look forward to being able to have one for longer, unfortunately if we did we would be very reluctant to part with it.

Introducing the personal computer you've waited for. THE EXIDY SORCERER.

SORCERER COMPUTER SYSTEM *Complete with Monitor*

The Sorcerer Computer is a completely assembled and tested computer system. Standard configuration includes 63-key typewriter-style keyboard and 16-key numeric pad, Z80 processor, dual cassette I/O with remote computer control at 300 and 1200 baud data rates, RS232 serial I/O for communications, parallel port for direct Centronics printer attachment, 4K ROM operating system, 8K ROM Microsoft BASIC In Rom Pac™ cartridge, composite video of 64 char/line 30 line/screen, 128 upper/lower case ASCII set and 128 user-defined graphic symbols, operation manual, BASIC programming manual and cassette/video cables, connection for S-100 bus expansion.

only £950 Credit facilities available.
+ 8% VAT

LOOK!
- 32K RAM on board
- RS232 interface
- 8K BASIC ROM
- CUTS interface
- 4K MONITOR
- KANSAS CITY interface
- S100 BUS
- User defined graphic symbols
- Z80 cpu

Teleplay presents the PROGRAMAGAME of all time!

TELEPLAY have produced a new product of easy to assemble cartridge games, for you and all your family, that will not go out of date. Just ADD each new cartridge, as it becomes available, and your colour video centre will be up to date the Teleplay way. You will save pounds in the long run. Works OK on Black and White TV.

EX STOCK IMMEDIATE DELIVERY

REALISTIC SOUND EFFECTS

We offer all this to you for ONLY **£28·90** + £2 31 VAT
Fully Assembled £35 90

The Teleplay Kit consists of a Professional Finished Inject Moulded Box Joystick Hand Controls – one cartridge which consists of 10 games – Football, Tennis, Solo Squash (1 & 2 players), Hockey, Basketball (1 & 2 players), Shoot etc. Mains Adaptor All holes are pre drilled No Special equipment required. You'll be surprised how easy it is to assemble this professional kit! Instruction Manual supplied separately at 0.45p. TELEPLAY GIVES YOU TWICE THE FUN BUILDING AND PLAYING THE MOST PROFESSIONAL TV GAME KIT EVER OFFERED.

Main Console has 3 Facility Handicaping Switches

Every 3 months a NEW game will become available to you and those already under production include Submarines, Space War, Wipe Out and many more. The price of these will vary from £12 to £19 depending upon the complexity of the game.

Additional cartridges NOW available include
Stunt Cycle – 4 games — Cartridges and Hand Controls £14.90 + £1 14 VAT
Tank Battle — Cartridge and Hand Controls £18.90 + £1 51 VAT
Road Race 2 games — Cartridge — Controls already with basic kit – £12.90 + £1 03 VAT
Fully assembled add £3.00

Cartridges come complete with printed case and easy to follow assembly instructions

NEW– WIPE OUT cartridge – 6 games £13.90 + £1.11 VAT

All parts fully guaranteed

Electrical knowledge is not a necessity to assemble this project just simple soldering.

KEY BOARD

756 GEORGE RISK

Brand new professional ASCII keyboards (USA)
Full technical details included. RRP £60.00
Only **£49.90** + 8% VAT.
Ready built, tested and guaranteed.

INTERESTED IN HOME COMPUTING?

FREE B BUG valued at £23.00 plus 10 x C12 cassettes valued at £4.00 WITH EVERY NASCOM

Start now and don't get left behind THE NASCOM 1 is here Ex-stock with full technical services

Plus the opportunity to join the fastest moving club of personal computer users enabling you to get the most our of your computer. You can OBTAIN and EXCHANGE programs and other software — many now available.

The Powerful Z80 Microprocessor
Professional Keyboard
1 Kbyte Monitor in EPROM
2 Kbyte RAM (expandable)
Audio Cassette interface
Plugs into your domestic TV
Easy construction from straightforward instructions
— no drilling or special tools
— Just neat soldering required.

Only £197.50 + 8% VAT (includes p & p + insurance)

Manuals seperately	2.95	Monitor quality improved TV Modulator	2.50
Z80 programming Manual	6.90		
Z80 Technical Manual	2.95		
PIO Technical Manual	2.95	Power supply suitable for NASCOM	19.90
(All prices add 8% VAT)			

NASCOM AD ONS — Nascom improved monitor B Bug (2K) featuring — *Four times tape speed *Direct text entry without ASCII *Extended keyboard facility *Additional useful subroutines £23.00

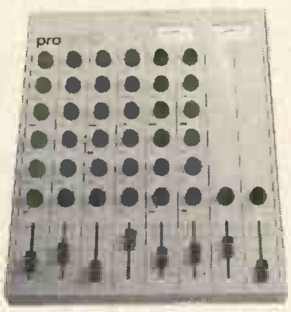
Nascom Software library. Send SAE for lists and prices.

COMPUTER JOYSTICK

Plugs into your Nascom P.I.O. No extras. Software and full documentation supplied. Plus free game cassette.
£14.90 each **£28·90** per pair

COMP PRO Mixer

Professional audio mixer that you can build yourself and save over £100.



6 into 2 with full equalization and echo, cve and pan controls.

All you need for your own recording studio is a stereo tape or cassette recorder.

This superb mixer kit has slider faders, level meters and additional auxilliary inputs.

Only **£99.90** plus 8% VAT for **complete kit** Plus **FREE** power supply valued at £25.00

Ideal for
DISCOS STAGE MIXING HOME STUDIOS
AND MANY OTHER APPLICATIONS

COMPUTER COMPONENTS

Send for our Spring 1979 catalogue. 0.60p Full of Computer Components, Peripherals and systems.

BLANK C12 Racal Quality CASSETTES £4.00 for 10

All prices include VAT except where shown. Orders over £5 post and packing free otherwise add 20p. Please make cheques and postal orders payable to COMP, or phone your order quoting BARCLAYCARD or ACCESS number.

COMP COMPUTER COMPONENTS

14 STATION ROAD · NEW BARNET · HERTFORDSHIRE · TEL: 01-441 2922 (Sales) 01-449 6596
CLOSE TO NEW BARNET BR STATION — MOORGATE LINE
OPEN — 10am to 7pm — Monday to Saturday

PCW Book Review

Michael James

A STEP BY STEP INTRODUCTION TO 8080 MICROPROCESSOR SYSTEMS

by David L. Cohn and L. Melsa, Dec. 1977;
Dilithium Press, £5.70
Exclusive distributors: ISB Services (Europe),
8 William Way, Letchworth, Herts SG6 2HG Tel: 046 26 3742

With the rapid growth of the number of Personal Computer enthusiasts, coming from all walks of life, the supply of Personal Computer equipment has grown to meet their demand and, in many cases, to create it. This, obviously, has created much confusion as to what equipment one should acquire and more importantly has posed the question as to how microprocessors actually work.

So far, the number of microprocessor I/C's offered by Personal Computer manufacturers is limited and among the most popular ones are the 8080's: originally designed and manufactured by INTEL, but now also available from several other vendors.

In their preface to the book, the authors write, "This microprocessor book is written for people who don't know anything about microprocessors but who wish they did." Reading the book, I felt that they have achieved their goal in bringing the subject matter close to the beginner. The book is written in an engaging, easy-to-read style which introduces microprocessors, and then goes on to describe the 8080 architecture and instruction set through simple examples. At the end of every chapter there are exercises to help the reader.

In later chapters some basic software is introduced (e.g. monitors and editors); then programming languages like Assembler and PL/M; then I/O structure and interface devices. The book ends with appendices listing the 8080 instruction set both alphabetically and by function.

Although the book is mainly aimed at beginners in the field of microprocessors, it can also provide useful information for experienced people who want a comprehensive understanding of the 8080's architecture and programming.

V. Nicola

Programme Library 1. Analysis of Aircraft Structures.

S. Constant and E. Sanford
Second Edition, Feb. 1978. 120pp. £4.85
(I.C.M., 1 Crofton Court, Cypress Road, London SE25 4BB)

This is a homebrew book in the sense that it isn't in fancy print. But everything else about it is professional – and highly specialised. The book comprises programmes written for the Casio PROfX – 1 (magnetic card) calculator. It will prove useful to students and teachers because the authors provide both listings and notes. The end page quotes Pythagoras, "Numbers take Man by the hand and conduct him unerringly along the path of reason".

There are about forty programmes in the book covering such topics as the Castigliano Theorem, Cut-Out Analysis, Torsion-Link – Torsional Stiffness and Minimum Dimensions. Highly recommended but only for the specialist. **PCW**

FROM READER TO REVIEWER

Read any good books on personal computing lately? Would you like to send in a review? Though publication cannot be guaranteed, we would like to look at your reviews.

BRAINS v BRAWN

OLIVER DIXON

AN IRATE USER

The evolution of a computer program

PCW *This article is an example of what our magazine stands for. Ideas spark off ideas and readers may become authors. Really, there may be specialists in computing but there are no experts. All of us have something to teach and something to learn.* PCW

In a recent article (P.C.W., Sept. 1978), Derek Chown gives a program for the TI 57 for finding right-angled triangles with sides of integer length. He quotes the example of the 3 : 4 : 5 triangle which everybody knows, and the 5 : 12 : 13 triangle which is as far as most people's knowledge of the subject extends.

Treble Trouble

I, too, had wondered about the existence of other 'Pythagorean triplets' as they are known in the trade, and a little while ago, I wrote a program in BASIC and ran it on a Wang 2200B to see what other triplets (if any) there were to be found.

My method was broadly similar to Derek's in that I fed in two sides and tested the third to find out if it was an integer. One point of difference was that I fed in the two shortest sides, X and Y, and tested the longest side Z. Another difference was that whenever I did find such a triplet, I tested it to see whether it was simply a multiple of a triplet already known (e.g. 6 : 8 : 10) by dividing through the triplet by all numbers less than X in turn.

I quote the program (Program 1) as it is a textbook example of a highly inefficient use of computer capabilities. After running the program for 1½ hours, an irate user, who had a more legitimate claim on computer time than I had, interrupted the program. But not before the computer had run through more than 30,000 possible combinations of X and Y and unearthed no fewer than 38 sets of triplets; that is, all those in which both the shortest side X is less than 66 and the middle side is less than 501.

Having established that there were in fact many such triplets, I next got to thinking about streamlining the program.

One section of the program which lends itself to improvement is the procedure for testing the triplet for divisibility by an integer. The original method is inefficient, but as it is only implemented on those infrequent occasions when a triplet is found, it only increases the running time of the program by a very small percentage.

The problem is in fact one of finding whether the highest common factor (H.C.F.) of X, Y and Z is larger than 1 (in which case we can reject the triplet) or equal to 1 (in which case we can describe the members of the triplet as relatively or mutually prime, and add a new triplet to the list).

This can be solved by the famous Euclidean algorithm:

To find the H.C.F. of two numbers X and Y where $Y > X$:
$Y/X = I_1$, remainder R_1 where I_1 is an integer
We repeat with $X/R_1 = I_2$, remainder R_2
and $R_1/R_2 = I_3$, remainder R_3
down to $R_{n-2}/R_{n-1} = I_n$, remainder R_n.
When we reach the point at which we are left with the remainder $R_n = 0$ then R_{n-1} is the H.C.F. of X and Y.

This algorithm may be readily programmed for computer solution. It is best written as a **subroutine**, because the algorithm will only find the H.C.F. of two numbers at a time. So we first find the H.C.F. of X and Y. Let us call it N. If $N > 1$, we then call the subroutine again to find the H.C.F. of N and Z. If *that* is also more than 1, we reject that particular triplet.

Odds against

This, however, does nothing to shorten the enormous odds against finding a triplet. If we take as a benchmark, the number of combinations tested to find the first twenty triplets, we come up with over 13,000 or odds of about 660 to 1.

One very simple idea for improving the program is to alter the arrangement of the loops. If we rewrite:

```
20 FOR Y = 1 TO 500
30 FOR X = 1 TO Y
140 NEXT X
150 NEXT Y
```

we reduce our benchmark measurement to fewer than 7,500 passes for the first 20 triplets. Note that this will not be an identical set of triplets to those found by Program 1. It should also be noted that if we allow the program to run through to Y = 500, the odds will be no better than in Program 1 in the long run. The improvement in the early stages is due to the fact that combinations of small X and large Y (which are largely fruitless) are not tested until the program has been running for some time. The program as it now stands is listed as Program 2.

A further saving may be effected in the number of times the inner loop needs to be performed. Let us consider what happens to Z over a small portion of the run:

Y	X	Z	Z–Y
12	1	12.0416	0.0416
12	2	12.1655	0.1655
12	3	12.3693	0.3693
12	4	12.6491	0.6491
12	5	13.0000	1.0000
12	6	13.4164	1.4164
12	7	13.8924	1.8924

It is readily apparent that for the lowest values of X, the difference (Z–Y) will be less than 1. As Y is integer, it follows that for these low values of X, solutions of Z cannot possibly be integer. The limiting value of X for this condition is $\sqrt{(2*Y + 1)}$ which in this example is exactly 5.

The limiting value of X as derived from this formula is not usually an integer, as can be seen from the next example.

Y	X	Z	Z–Y
13	1	13.0384	0.0384
13	2	13.1529	0.1529
13	3	13.3417	0.3417
13	4	13.6015	0.6015
13	5	13.9284	0.9284
13	6	14.3178	1.3178
13	7	14.7648	1.7648

Here the lowest value of X for which Z can be integer lies between 5 and 6, or 5.1961 to be precise. We are, of course, no more interested in non-integer X than we are in non-integer Z. All we need to say is that an integer solution of Z is manifestly impossible *if X has an integer value of 5 or less.*

We can take advantage of this fact by rewriting

```
30 FOR X = INT(SQR(2*Y + 1) ) TO Y
```

This amendment effects a small economy of about 14% in the number of combinations searched, or in other words a benchmark value of about 6300.

Using new facts

Derek Chown's program is structured rather differently. He makes use of two facts about the triplets which I had overlooked: namely that the longest side, Z, is always an odd number; and that the shortest side, X, cannot be greater than $Z/\sqrt{2}$..

By reasoning similar to that above, it can be shown that for any given longest side, Z, there is a minimum length of shortest side, X, if Y is to be an integer. This shortest side is given by $X = \sqrt{(2*Z - 1)}$.

We can now put together a program which incorporates all we have so far discovered as Program 3. The benchmark value for the first 20 triplets in this case is a little over 2000.

However with odds of 100 to 1 against, it is still very much a 'blunderbuss' sort of program, firing off massive salvoes of pairs of sides in the hope that the third side will hit the target.

If we examine the list of triplets generated by Program 1 (Table 1), some sort of pattern can be seen. It is readily apparent that there is a whole family of triplets in which Y and Z differ by exactly 1; and that there is one such set of triplets for every odd value of X where X = 3 or more.

It can be easily shown that these triplets take the form:
X ; $Y = (X^2 - 1)/2$; $Z = (X^2 + 1)/2$ where X is any odd number greater than 1. Writing a program to generate this family of triplets is a very straightforward matter.

There are, however, still a lot of triplets unaccounted for, in which Y and Z differ by more than 1. The rules governing other 'families' of triplets in which (Z – Y) = 2,8,9 etc. are rather more complicated.

A more general solution

I wondered, therefore, if a yet more general solution existed. Upon investigation, I discovered that this is one of the classical problems of number theory (Reference 1). The general solution was published by Euclid (c.300 B.C.) and was probably known to the Babylonians more than a thousand years before that.

For any two integers A and B where A > B and one of them is odd and the other even, the terms $(A^2 - B^2)$, $2*A*B$ and $(A^2 + B^2)$ will form a Pythagorean triplet. Furthermore, if A and B are relatively prime, there will be no 'duplicates' which are simple multiples of other triplets.

We finish up, therefore, with Program 4 which will generate triplets to our heart's content faster than they can be printed out. A small sample of the output of this program is shown in Table 2. If we run the program up to A = 50, B = 49 we will have over 500 triplets which should be enough to satisfy anybody!

The moral of this story is to reinforce one of the basic axioms of programming: *every program can be made shorter and be made to run faster* or put another way *Brain beats Brawn every time.*

Ref. 1 Leveque, W. J. Elementary theory of numbers Addison-Wesley, 1962

3	4	5
5	12	13
7	24	25
8	15	17
9	40	41
11	60	61
12	35	37
13	84	85
15	112	113
16	63	65
17	144	145
19	180	181
20	21	29
20	99	101
21	220	221
23	264	265
24	143	145
25	312	313
27	364	365
28	45	53
28	195	197
29	420	421
31	480	481
32	255	257
33	56	65
36	77	85
36	323	325
39	80	89
40	399	401
44	117	125
44	483	485
48	55	73
51	140	149
52	165	173
57	176	185
60	91	109
60	221	229
65	72	97

Table 1

A	B	X	Y	Z
2	1	3	4	5
3	2	5	12	13
4	1	15	8	17
4	3	7	24	25
5	2	21	20	29
5	4	9	40	41
6	1	35	12	37
6	5	11	60	61
7	2	45	28	53
7	4	33	56	65
7	6	13	84	85
8	1	63	16	65
8	3	55	48	73
8	5	39	80	89
8	7	15	112	113
9	2	77	36	85
9	4	65	72	97
9	8	17	144	145
10	1	99	20	101
10	3	91	60	109
10	7	51	140	149
10	9	19	180	181
11	2	117	44	125
11	4	105	88	137
11	6	85	132	157
11	8	57	176	185
11	10	21	220	221
12	1	143	24	145
12	5	119	120	169
12	7	95	168	193
12	11	23	264	265
13	2	165	52	173
13	4	153	104	185
13	6	133	156	205
13	8	105	208	233
13	10	69	260	269
13	12	25	312	313
14	1	195	28	197
14	3	187	84	205
14	5	171	140	221
14	9	115	252	277
14	11	75	308	317
14	13	27	364	365
15	2	221	60	229
15	4	209	120	241
15	8	161	240	289
15	14	29	420	421
16	1	255	32	257
16	3	247	96	265
16	5	231	160	281
16	7	207	224	305
16	9	175	288	337
16	11	135	352	377
16	13	87	416	425
16	15	31	480	481
17	2	285	68	293
17	4	273	136	305
17	6	253	204	325
17	8	225	272	353
17	10	189	340	389
17	12	145	408	433
17	14	93	476	485
17	16	33	544	545
18	1	323	36	325
18	5	299	180	349
18	7	275	252	373
18	11	203	396	445
18	13	155	468	493
18	17	35	612	613
19	2	357	76	365
19	4	345	152	377
19	6	325	228	397

Table 2

```
10 REM THIS IS PROGRAM"PYTRIP1" FOR
   FINDING PYTHAGOREAN TRIPLETS
15 PRINT "    X       Y       Z"
20 FOR X=1 TO 500
30 FOR Y=X TO 500
40 Z=SQR(X!2+Y!2)
50 IF INT(Z)[]Z THEN 140
55 REM TEST FOR DIVISIBILITY
60 I=1
62 I=I+1
70 IF INT(X/I)=X/I THEN 90
80 GOTO 120
90 IF INT(Y/I)=Y/I THEN 110
100 GOTO 120
110 IF INT(Z/I)=Z/I THEN 140
120 IF I[X THEN 62
130 PRINTUSING 135,X,Y,Z
135% ####  ####  ####
140 NEXT Y
150 NEXT X
160 STOP
```

Programme 1

```
10 REM THIS IS PROGRAM "PYTRIP3" FOR
   FINDING PYTHAGOREAN TRIPLETS
15 PRINT "    X       Y       Z"
20 FOR Z=3 TO 500 STEP 2
30 FOR X=INT(SQR(2*Z-1)) TO Z/SQR(2)
40 Y=SQR(Z!2-X!2)
50 IF INT(Y)[]Y THEN 140
60 M=Y:N=X
70 GOSUB 200
80 IF N=1 THEN 130
90 M=Z
100 GOSUB 200
110 IF N[]1 THEN 140
130 PRINTUSING 135,X,Y,Z
135% ####  ####  ####
140 NEXT X
150 NEXT Z
160 STOP
200 REM SUBROUTINE FOR FINDING H.C.F.
210 R=M-INT(M/N)*N
220 IF R=0 THEN 260
230 M=N
240 N=R
250 GOTO 210
260 RETURN
```

Programme 3

```
10 REM THIS IS PROGRAM "PYTRIP2" FOR
   FINDING PYTHAGOREAN TRIPLETS
15 PRINT "    X       Y       Z"
20 FOR Y=1 TO 500
30 FOR X=1 TO Y
40 Z=SQR(X!2+Y!2)
50 IF INT(Z)[]Z THEN 140
60 M=Y:N=X
70 GOSUB 200
80 IF N=1 THEN 130
90 M=Z
100 GOSUB 200
110 IF N[]1 THEN 140
130 PRINTUSING 135,X,Y,Z
135% ####  ####  ####
140 NEXT X
150 NEXT Y
160 STOP
200 REM SUBROUTINE FOR FINDING H.C.F.
210 R=M-INT(M/N)*N
220 IF R=0 THEN 260
230 M=N
240 N=R
250 GOTO 210
260 RETURN
```

Programme 2

```
10 REM THIS IS PROGRAM "PYTRIP4" FOR
FINDING PYTHAGOREAN TRIPLES THE EASY WAY
15 PRINT "  A    B       X       Y       Z"
20 FOR A=2 TO 50
25 IF INT(A/2)=A/2 THEN 27
26 B1=2:GOTO 30
27 B1=1
30 FOR B=B1 TO A-1 STEP 2
40 M=A:N=B
50 GOSUB 200
90 IF N]1 THEN 160
100 X=A!2-B!2
110 Y=2*A*B
120 Z=A!2+B!2
150 PRINTUSING 155 ,A,B,X,Y,Z
155% ##   ##     ####   ####   ####
160 NEXT B
170 NEXT A
180 STOP
200 REM SUBROUTINE FOR FINDING H.C.F.
210 R=M-INT(M/N)*N
220 IF R=0 THEN 260
230 M=N
240 N=R
250 GOTO 210
260 RETURN
```

Programme 4

What Shall We Do In School Today

Charles Sweeten

People from this country who have visited the States recently, have noticed that there is a vital difference between the countries in the way they use computers in education. In this country, by and large we use computers to teach children about computers. In the States they use computers to teach children about everything but computers. The reasons for this may never be known, but I suspect it has something to do with our worship of the examination result, and their preoccupation with social issues.

Cutbacks

Most schools in this country have suffered from cutbacks in expenditure, and as a result are short of many of the necessities of teaching. In this situation it has been difficult for even the most enthusiastic teacher to get the LEA to provide an in-school micro. Fortunately other pressures bear on the situation. There are quite a lot of parents these days who realise that computers are going to be fundamental to the society of our children and are very anxious that their school should recognise this and be doing something about it. They are often only too glad to help raise money for a computer rather than a mini-bus and will require no persuasion to bring pressure to bear on both the Headteacher and the LEA. Parent teacher meetings with the LEA advisor with special responsibility for computing, and one exists in every authority, can be most entertaining. The Headteacher needs little encouragement to be sure; but he lacks funds. However he is usually susceptible to the argument that a computer in the school will bring prestige, and sometimes the more interesting pupil. The best argument of all that can be applied to a school that has already got a terminal is one of cost against usage. Unfortunately the LEA's have too much money already invested in expensive mainframes to listen to the true figures. While so few, relatively, of our schools have any sort of computing at all, it is a pity when one hears very responsible people saying that hardware is no problem, and that anyone can get it themselves. They have heard of schools getting fed up with receiving no help and no funds, and then they have heard of those schools helping themselves and raising money for equipment, and this sounds like too good an excuse for the LEA's to spend money on something else.

Respectability

Those schools that have been successful in getting computing have usually gone along the following course. First they set up an arrangement with the local College providing computer services to the LEA whereby they take mark sense cards for processing, having embarked on a CSE or O level in Computer Science. This makes the whole thing "respectable", and occupies a group of pupils, so that it does not require further staffing. At some stage the whole course gets out of hand; the pupils get too enthusiastic; there are too many pupils taking the course; they know more than the teacher; and you discover something that nobody ever tells you, which is that the batch collection is on Monday, your class is on Tuesday and Thursday, and the cards are returned on Friday. At this stage the LEA announce their new 2 million pound computer and justifies it by putting you on a terminal link, but making you pay the phone bill. Over the next two years you overspend your phone bill allowance by between 4 and 10 times, and you still can't get 24 simple programs processed in under two weeks. In desperation you raise the money for a micro on condition that you stop using the terminal link. You think this will solve your problems. Actually it won't. You will find that the number of people who want to use it will go up by ten times and the bottleneck is worse than before. You will however get enormous fun in the process. But are you doing the right courses anyway, I ask. I know you justified the whole thing on the basis of exam course, but have you thought about the alternatives?

We are left with the situation that in order to justify successfully the case for computing in our schools, we are forced to introduce a course of Computer Science. It may of course be right to do so in some cases, but many of the schools trying to achieve the same end now, may be misled into thinking that this is the right means, without having considered the problem for themselves. It is this that I particularly want to talk about.

Let me state six principles, without justification, on which to base our pattern of computer use within a school.

PRINCIPLES

1. All pupils should know what a computer is and how it is organised.
2. They should know what a program is, and why it is used.
3. They should know when it is useful to use a computer and when it is not.
4. They should be aware of data storage in commerce and administration, and of simulation and modeling.
5. They should be aware of the need for documentation at all stages of problem solving.
6. Some pupils will need to study the theory or the applications in detail.

It is easy to disagree with principles and I am in sympathy with anyone doing so, but you must replace them and apply those principles to your own case.

I now want to suggest a choice of courses that we could put on in our schools, and I believe that all of us should try to think which are the most suitable for our own situation. I want to get away from the situation where we are forced into one particular line because that is the only line on which we can argue our case to the authorities.

COURSES

1. An introduction for all 13 year olds along the lines of the first three principles.
2. The beginnings of applications for 14 year olds. For example the Local History Projects at Ipswich, and some examples which are available in Geography.
3. An appreciation course for 16+ pupils covering principles four and five which is aimed at the pupil who would not otherwise meet this material.
4. A specialised course leading to CSE or O level.
5. A specialised course leading to A level or City and Guilds.
6. Use of packages and models in A-level Mathematics.
7. Applications of models and simulations in various subjects at A-level. For example the Chelsea Science Simulation Packages.
8. The deliberate encouragement of the extended project, which requires staffing and the designing of a model.
9. The construction of computer models under the general heading of electronics.

Fashion

Ten years ago people all round me were crying that appreciation courses for all were the thing, since we would all be affected by computers. Privacy, data banks, the reduction of everyone to numbers, and the impossibility of arguing with a computer, seemed possible dangers and children had to be prepared. Success at implementing such schemes was rare, but our reasoning has been proved correct in those areas where large computers exist. One has only to think of the vehicle licensing centre to see this. Then came the flight into Computer Science as we struggled to get a sense of direction and failed to get the computing facilities we really needed. But surely we can now see the new direction.

Yes, computers will affect us. But not remotely as we thought then. It is in a far more personal way. Watches, washing machines, door locks, ovens, are here now. Everything that works has the potential for computer control. And control means programming. What we have to teach is that programs have to be written; that programs are no better than their authors; and that programs can be altered to suit our needs better. This leads to the need to be able to specify our problems in the first place. We will all need to be systems analysts up to a point.

But what we need is more than this. In secondary education we must provide the children with the building methods. We cannot connect everything that moves onto a micro-processor. Take the British Economy for example (there are those who claim that it moves). Do our pupils know what a model of this means? Start talking about the route for the channel tunnel and it will be at least 20 seconds before someone says get a computer to "work it out", whatever "it" is. The philosophy is quite simple. You state a problem; you make a suggestion; you make a model on this basis; you get some "results". And then you ask —

DOES IT WORK? IS IT BEAUTIFUL?

This surely is what we should be trying to teach. We have suffered enough from that prolonged state of childhood where a person worships ideas merely because they are written down, or results merely because they are given as numbers with many digits.

And so I have come to regard the study of Hollerith, two's complement arithmetic, the importance of a mask, the handshake principle, and all their kind, as irrelevant to the vast majority of schools. By all means teach this to the few, but if that is all we can do then we should be ashamed. We must apply the computer to all aspects of learning where conjecture and testing take place, because this is what will be happening in the real world outside schools.

It is vital therefore that we should redouble our efforts to involve teachers from other departments. Teacher training institutions must train their students in where it is appropriate to use the computer. School children must get involved in the modelling process as early as possible and must learn to ask "does it work? Is it beautiful?". I hope I should not have to say which of the above courses I regard as essential, and which I regard as a specialist option for some children. Nor, I hope, should I have to say that if our resources are thin, then our priorities must lie with the majority.

Talking of resources, I am all too aware of the cutbacks in expenditure this year, but there are still many institutions coming to MUSE and asking us what we feel they should have in the way of micro-computer facilities. In particular, I am often asked "what computer should I buy". The answer is not that simple — it all depends on what you want it to do. Let's take the courses one by one.

Temptation

For the junior introductory course you must avoid the temptation to teach programming. You need to be able to demonstrate machine code, assembler code, a high level language, backing storage and the ability to attach anything to the I/O ports. For this I think you need a disk system, a simplified display of memory contents, and easy to use I/O devices. If you argue that this is quite unnecessarily lavish, then I agree with you — you don't need anything at all. But if you *are* going to show a system then choose one that does what you need.

For the junior applications you will need the ability to load programs in BASIC quickly and in some cases you will need to use a data base. This is the cheapest system of all to buy if you can do without the disk. It is essential to have a large monitor available for the class to see, but the system need cost little more than 800 pounds. The 16+ appreciation course has the same requirements.

The course leading to CSE or O level needs BASIC and a lower level language. It does not need a disk, but a decent casette handling system is an advantage. An ability to come out of BASIC is essential.

For the A-level course the front panel mode of the 380Z is ideal but you need a disk to deal with the concepts of file handling and operating systems. It is also essential to be able to handle Editor, Assembler, and various high level languages easily and efficiently, and the SWTPC system is ideal in this respect.

For the A-level Maths course you need a disk system to be able to call up any package or program quickly during a period. A further advantage would be, easy to use and comprehensive graphics. These are at present only available on the Apple. Graphics are available on other systems, but they do not begin to compare with the Apple graphics.

In other A level subjects it is perhaps not yet necessary to have disks, as the packages available are rather lengthy and it is unlikely that a teacher will want to get in more than one program in a period. The packages that have been released so far do not make use of graphics and so a simple system will suffice. But if you feel that you might write some material of your own, then you must be interested in graphics.

For projects of course you will need the biggest, fastest, and most complicated system that you can find, and it will still be inadequate to cope with the demands made on it. It is perhaps worth making a virtue out of managing on less than you need.

I would strongly advise anyone who is thinking of constructional projects, other than the initial assembly of a kit into your working system, to keep the work separate. Construction has a habit of taking the system out of commission just when you need it most. And if you are thinking of buying a Nascom, for example, and building it into a school computer then excellent; but buy a computer as well to actually do the work, because construction and use are two different things at the school level.

Now this is all very well, but will the authorities allow us to get on and do the job? Who are the authorities and who actually takes the decisions that matter? For most of us the LEA is the final arbiter. But who makes the policies that they follow? If you can find ten teachers who are involved in the actual decision making then please publish the list as we would all like to meet them.

Decision

So what I am saying is this. We should decide in the first place the part that computers have to play in education, and that decision must be to involve the whole teaching process in all subjects. The computer is the modelling tool of the future and the tool is here now. Computer Assisted Instruction (CAI) is not just an American alternative to real education, nor does it involve vast expense, it is available now on cheap micro-computers and it really is useful. Having decided that, we must decide what courses will do the most good in our particular schools, bearing in mind the money available and the teachers available, and the courses are the ones that involve the ordinary teaching of all pupils.

Let me end by giving two examples from the classroom.

The first is from A-level Statistics. I want to be able to set up a model and examine it by taking statistics from a data bank. I want to be able to stand in front of the class and ask them what to do. I want to be able to take samples and group them in any way they desire, process the data in any way they want, and I need to be able to call up the relevant programs at a second's notice. They must be able to call for any statistical test that they need and the results of their investigations must be immediately available to them. Any method and any data bank must available. Sounds nice? Well, it is available now.

The second is from History. I would like to give a lagging pupil extra work on events between 1910 and 1920.

Computer:	What major events took place in this period?
Pupil:	The Bolshoi revolsion ... I think
Computer:	Good, but watch your spelling. The Bolshevik Revolution. Did you forget one other event?
Pupil:	The Boer War
Computer:	Well, it certainly was a war, but not that one. Try again.
Pupil:	World War 1
Computer:	Good. Can you give me the dates as well?
Pupil:	The 14 - 18 war.
etc.	

Well, you can have this as well for under £100 on top of a standard micro-computer.

What shall we do in school today? Use the tool of tomorrow.

(For information about micro-computers in education, write to the Secretary of MUSE, 18 South Road, Oundle, Peterborough.)

DIODES/ZENERS

1N914	100v	10mA	.05
1N4005	600v	1A	.08
1N4007	1000v	1A	.15
1N4148	75v	10mA	.05
1N4733	5.1v	1 W Zener	.25
1N753A	6.2v	500 mW Zener	.25
1N758A	10v	"	.25
1N759A	12v	"	.25
1N5243	13v	"	.25
1N5244B	14v	"	.25
1N5245B	15v	"	.25

SOCKETS/BRIDGES

8-pin	pcb .20		ww	.35
14-pin	pcb .20		ww	.40
16-pin	pcb .20		ww	.40
18-pin	pcb .25		ww	.75
22-pin	pcb .35		ww	.95
24-pin	pcb .35		ww	.95
28-pin	pcb .45		ww	1.25
40-pin	pcb .50		ww	1.25
Molex pins .01		To-3 Sockets		.25
2 Amp Bridge		100-prv		.95
25 Amp Bridge		200-prv		1.95

TRANSISTORS, LEDS, etc.

2N2222	NPN (2N2222 Plastic .10)	.15
2N2907	PNP	.15
2N3906	PNP (Plastic - Unmarked)	.10
2N3904	NPN (Plastic - Unmarked)	.10
2N3054	NPN	.35
2N3055	NPN 15A 60v	.50
T1P125	PNP Darlington	.35
LED Green, Red, Clear, Yellow		.15
D.L.747	7 seg 5/8" High com-anode	1.95
MAN72	7 seg com-anode (Red)	1.25
MAN3610	7 seg com-anode (Orange)	1.25
MAN82A	7 seg com-anode (Yellow)	1.25
MAN74A	7 seg com-cathode (Red)	1.50
FND359	7 seg com-cathode (Red)	1.25

C MOS

4000	.15
4001	.15
4002	.20
4004	3.95
4006	.95
4007	.20
4008	.75
4009	.35
4010	.35
4011	.20
4012	.20
4013	.40
4014	.75
4015	.75
4016	.35
4017	.75
4018	.75
4019	.35
4020	.85
4021	.75
4022	.75
4023	.20
4024	.75
4025	.20
4026	1.95
4027	.35
4028	.75
4030	.35
4033	1.50
4034	2.45
4035	.75
4040	.75
4041	.69
4042	.65
4043	.50
4044	.65
4046	1.25
4049	.45
4050	.45
4066	.55
4069/74C04	.25
4071	.25
4081	.30
4082	.30
MC 14409	14.50
MC 14419	4.85
4511	.95
74C151	1.90

– T T L –

7400	.10
7401	.15
7402	.15
7403	.15
7404	.10
7405	.25
7406	.25
7407	.55
7408	.15
7409	.15
7410	.15
7411	.25
7412	.25
7413	.25
7414	.75
7416	.25
7417	.40
7420	.15
7426	.25
7427	.25
7430	.15
7432	.20
7437	.20
7438	.20
7440	.20
7441	1.15
7442	.45
7443	.45
7444	.45
7445	.65
7446	.70
7447	.70
7448	.50
7450	.25
7451	.25
7453	.20
7454	.25
7460	.40
7470	.45
7472	.40
7473	.25
7474	.30
7475	.35
7476	.40
7480	.55
7481	.75
7483	.75
7485	.55
7486	.25
7489	1.05
7490	.45
7491	.70
7492	.45
7493	.35
7494	.75
7495	.60
7496	.80
74100	1.15
74107	.25
74121	.35
74122	.55
74123	.35
74125	.45
74126	.35
74132	.75
74141	.90
74150	.85
74151	.65
74153	.75
74154	.95
74156	.70
74157	.65
74161	.55
74163	.85
74164	.60
74165	1.10
74166	1.25
74175	.80
74176	.85
74180	.55
74181	2.25
74182	.75
74190	1.25
74191	.95
74192	.75
74193	.85
74194	.95
74195	.95
74196	.95
74197	.95
74198	1.45
74221	1.00
74367	.75
75108A	.35
75491	.50
75492	.50
74H00	.15
74H01	.20
74H04	.20
74H05	.20
74H08	.35
74H10	.35
74H11	.25
74H15	.45
74H20	.25
74H21	.25
74H22	.40
74H30	.20
74H40	.25
74H50	.25
74H51	.25
74H52	.15
74H53J	.25
74H55	.20
74H72	.35
74H101	.75
74H103	.55
74H106	.95
74L00	.25
74L02	.20
74L03	.25
74L04	.30
74L10	.20
74L20	.35
74L30	.45
74L47	1.95
74L51	.45
74L55	.65
74L72	.45
74L73	.40
74L74	.45
74L75	.55
74L93	.55
74L123	.85
74S00	.35
74S02	.35
74S03	.25
74S04	.25
74S05	.35
74S08	.35
74S10	.35
74S11	.35
74S20	.25
74S40	.20
74S50	.20
74S51	.25
74S64	.15
74S74	.35
74S112	.60
74S114	.65
74S133	.40
74S140	.55
74S151	.30
74S153	.35
74S157	.75
74S158	.30
74S194	1.05
74S257 (8123)	1.05
74LS00	.20
74LS01	.20
74LS02	.20
74LS04	.20
74LS05	.25
74LS08	.25
74LS09	.25
74LS10	.25
74LS11	.25
74LS20	.20
74LS21	.25
74LS22	.25
74LS32	.25
74LS37	.25
74LS38	.35
74LS40	.30
74LS42	.65
74LS51	.35
74LS74	.35
74LS86	.35
74LS90	.55
74LS93	.55
74LS107	.40
74LS123	1.00
74LS151	.75
74LS153	.75
74LS157	.75
74LS164	1.00
74LS193	.95
74LS367	.75
74LS368	.65

LINEARS, REGULATORS, etc.

MCT2	.95
8038	3.95
LM201	.75
LM301	.45
LM308 (Mini)	.95
LM309H	.65
LM309K (340K-5)	.85
LM310	.85
LM311D (Mini)	.75
LM318 (Mini)	1.75
LM320K5 (7905)	1.65
LM320K12	1.65
LM320T5	1.65
LM320T12	1.65
LM320T15	1.65
LM324N	1.25
LM339	.75
7805 (340T5)	.95
LM340T12	.95
LM340T15	.95
LM340T18	.95
LM340T24	.95
LM340K12	1.25
LM340K15	1.25
LM340K18	1.25
LM340K24	1.25
78L05	.75
78L12	.75
78L15	.75
78M05	.75
LM373	2.95
LM380 (8-14 PIN)	.95
LM709 (8, 14 PIN)	.25
LM711	.45
LM723	.40
LM725N	2.50
LM739	1.50
LM741 (8-14)	.25
LM747	1.10
LM1307	1.25
LM1458	.65
LM3900	.50
LM75451	.65
NE555	.35
NE556	.85
NE565	.95
NE566	1.25
NE567	.95

9000 SERIES

9301	.85	95H03	1.10
9309	.35	9601	.20
9322	.65	9602	.45

MICRO'S, RAMS, CPU'S, E-PROMS

74S188	3.00	8214	8.95
1702A	4.50	8224	3.25
MM5314	3.00	8228	6.00
MM5316	3.50	8251	8.50
2102-1	1.45	8255	10.50
2102L-1	1.75	8T13	1.50
2114	9.50	8T23	1.50
TR1602B	3.95	8T24	2.00
TMS 4044-	9.95	8T97	1.00
		2107B-4	4.95
8080	8.95	2708	9.50
8212	2.95	Z80 PIO	8.50

INTEGRATED CIRCUITS UNLIMITED

7889 Clairemont Mesa Blvd., San Diego, CA 92111 U.S.A.

No Minimum

All prices in U.S. dollars. Please add postage to cover method of shipping. Orders over $100 (U.S.) will be shipped air no charge. Payment should be submitted with order in U.S. dollars. All IC's Prime/Guaranteed. All orders shipped same day received.

SPECIAL DISCOUNTS

Total Order	Deduct
$35 - $99	10%
$100 - $300	15%
$301 - $1000	20%

Phone (714) 278-4394 BarclayCard / Access / American Express / BankAmericard / Visa / MasterCharge

COMPARE

John Coll

This routine enables a file on disk to be compared with a file resident in memory. The address and contents of any section of the program which do not match are listed. This program will not check text files.

The general syntax of the COMPARE command is:

COMPARE, (filename)
A few examples follow:

+++ COMPARE BASIC
+++ COMPARE O.TEST.BIN

The default extension is .CMD and the default drive is the working drive.

This routine is useful for comparing two versions of the same program to list the changes, or to determine how running a program has changed it, if at all.

Example:

```
At  $0010 found  $A0  should be $00
At  $0011 found  $4A  should be $00
At  $0012 found  $A0  should be $30
At  $0013 found  $16  should be $0A
At  $16A3 found  $1B  should be $15
At  $16A4 found  $36  should be $C4
Comparison completed.
```

```
                        NAM     COMPARE
                        OPT     PAG,NOG
7600                    ORG     $7600
               *
               *JOHN A. COLL 16-6-78
               *
               *EQUATES
7103           WARMS    EQU     $7103
7112           PUTCHR   EQU     $7112
7118           PSTRNG   EQU     $7118
711E           PCRLF    EQU     $711E
7127           FILSPC   EQU     $7127
712D           SETEXT   EQU     $712D
7139           OUTHEX   EQU     $7139
7806           FMS      EQU     $7806
713C           RPTERR   EQU     $713C
7253           PDATA    EQU     $7253
7740           FCB      EQU     $7740
               *
7600 20 01     START    BRA     BEGIN
7602 01        VERSION  FCB     1
7603 CE 77 40  BEGIN    LDX     #FCB
7606 BD 71 27           JSR     FILSPC
7609 25 4F              BCS     ERR
760B 86 02              LDA A   #2          .CMD
760D CE 77 40           LDX     #FCB
7610 BD 71 2D           JSR     SETEXT
7613 CE 77 40           LDX     #FCB
7616 86 01              LDA A   #1          READ
7618 A7 00              STA A   X
761A BD 78 06           JSR     FMS         OPEN
761D 26 3B              BNE     ERR
761F 86 FF              LDA A   #$FF
7621 A7 3B              STA A   59,X        BINARY
7623 8D 2C     ALPHA    BSR     INBYTE
7625 81 02              CMP A   #2
7627 26 FA              BNE     ALPHA
7629 8D 26              BSR     INBYTE
762B B7 76 CE           STA A   MSB
762E 8D 21              BSR     INBYTE
7630 B7 76 CF           STA A   LSB
7633 8D 1C              BSR     INBYTE
7635 16                 TAB                 BYTE COUNT
7636 FE 76 CE           LDX     MSB
7639 09                 DEX
763A FF 76 CE           STX     MSB
763D 37        BETA     PSH B
763E 8D 11              BSR     INBYTE
7640 FE 76 CE           LDX     MSB
7643 08                 INX
7644 FF 76 CE           STX     MSB
7647 A1 00              CMP A   X
7649 26 23              BNE     SHOW
764B 33        GAMMA    PUL B
764C 5A                 DEC B
764D 26 EE              BNE     BETA
764F 20 D2              BRA     ALPHA
```

```
COMPARE                                          TSC MNEMONIC ASSEMBLER

 7651 CE 77 40  INBYTE  LDX     #FCB
 7654 BD 78 06          JSR     FMS
 7657 26 01             BNE     ERR
 7659 39                RTS
 765A A6 01     ERR     LDA A   1,X
 765C 81 08             CMP A   #8
 765E 27 06             BEQ     HOME
 7660 BD 71 3C          JSR     RPTERR
 7663 7E 71 03  FLEX    JMP     WARMS
 7666 CE 76 B8  HOME    LDX     #DONETX
 7669 BD 71 18          JSR     PSTRNG
 766C 20 F5             BRA     FLEX
 766E B7 76 CD  SHOW    STA A   DISK
 7671 CE 76 9D          LDX     #ATTXT
 7674 BD 71 18          JSR     PSTRNG
 7677 CE 76 CE          LDX     #MSB
 767A BD 71 39          JSR     OUTHEX
 767D CE 76 CF          LDX     #LSB
 7680 BD 71 39          JSR     OUTHEX
 7683 CE 76 A2          LDX     #FNDTXT
 7686 BD 72 53          JSR     PDATA
 7689 FE 76 CE          LDX     MSB
 768C BD 71 39          JSR     OUTHEX
 768F CE 76 AB          LDX     #DISKTX
 7692 BD 72 53          JSR     PDATA
 7695 CE 76 CD          LDX     #DISK
 7698 BD 71 39          JSR     OUTHEX
 769B 20 AE             BRA     GAMMA
 769D 41        ATTXT   FCC     /At $/
 76A1 04                FCB     4
 76A2 20        FNDTXT  FCC     / found $/
 76AA 04                FCB     4
 76AB 20        DISKTX  FCC     / should be $/
 76B7 04                FCB     4
 76B8 43        DONETX  FCC     /Comparison completed/
 76CC 04                FCB     4
 76CD           DISK    RMB     1
 76CE           MSB     RMB     1
 76CF           LSB     RMB     1
                        END     START

NO ERROR(S) DETECTED


 SYMBOL TABLE:

ALPHA   7623   ATTXT  769D   BEGIN  7603   BETA    763D   DISK   76CD
DISKTX  76AB   DONETX 76B8   ERR    765A   FCB     7740   FILSPC 7127
FLEX    7663   FMS    7806   FNDTXT 76A2   GAMMA   764B   HOME   7666
INBYTE  7651   LSB    76CF   MSB    76CE   OUTHEX  7139   PCRLF  711E
PDATA   7253   PSTRNG 7118   PUTCHR 7112   RPTERR  713C   SETEXT 712D
SHOW    766E   START  7600   VERSIO 7602   WARMS   7103
```

STATPACK

Colin Chatfield — Micro-Aid

Part 2 of this series of statistical programs includes an information program and a small practical program for moving averages which is section 4. Part 3 will include a program for basic statistics covering many averages.

STAT2 can be CHAINed from STAT1 which was published in the September issue or can be entered and run independently. The purpose is to remind you of the functions of the programs available in the series as it is all too easy to forget. So I suggest that you do not miss it out. Also for those with the facility any other part of the series can be CHAINed directly which is a tidy way of getting and running another program.

Part 1 was the data entry and verification program and part 2 is the information program. The other parts

```
0005 REM - STAT2 STATPACK INFORMATION PROGRAM
0020 ? CHR$(25);:?CHR$(25);:?CHR$(22);:?CHR$(12);
0080 LINE= 80
0100 ? TAB(20);"STAT2  -  HELP PROGRAM"
0180 ? "STATPACK IS USED BY ENTERING DATA AT THE TERMINAL ROW"
0190 ? "BY ROW USING STAT1 PROGRAM. YOU MUST STATE HOW MANY ITEMS THERE"
0200 ? "ARE PER ROW AND PER COLUMN. THE NUMBER OF ITEMS WILL DEPEND ON"
0210 ? "HOW MUCH MEMORY YOU HAVE. SAVED DATA CAN BE STORED ON DISC AND"
0220 ? "USED OVER AND OVER BY DIFFERENT PROGRAMS."
0230 ? :INPUT"      PRESS RETURN WHEN READY ",A$
0240 ? :?"YOU CAN DO VARIOUS CALCULATIONS AS FOLLOWS:":?
0270 ? "STATISTICS"," ","BASIC",3
0280 ? "AVERAGES","MEANS"," ",3
0282 ? " "," ","GEOMETRIC",3
0284 ? " "," ","HARMONIC",3
0300 ? " "," ","MEDIAN",3
0320 ? " "," ","MOVING",4
0330 ? " "," ","PROGRESSIVE",5
0332 ? "OTHER BASICS"," ","STD. DEVIATION",3
0334 ? " ","COEFFICIENT OF VARIATION",3
0335 ? " "," ","HIGH-LOW",3
0336 ? " "," ","VARIANCE",3
0338 ? " "," ","RANGE",3
0340 ? "CHI-SQUARE"," "," ",6
0350 ? "BAR CHARTS"," "," ",7
0400 ? "SPARE PROGRAM"," "," ",8
0405 ? "RELATIONSHIPS"," "," ",9
0410 ? "REGRESSION"," ","COEFFICIENT OF",10
0420 ? " "," ","CORRELATION",10
0425 ? " "," ","LINEAR",10
0430 ? :?"NEW DATA ENTRY"," "," ",1
0435 ? "INFORMATION"," "," ",2:?
0440 ? :INPUT"     ENTER PROGRAM NUMBER YOU REQUIRE ",A
0450 ON A GOTO 510,20,530,540,550,560,570,580,590,600
0510 CHAIN STAT1
0530 CHAIN STAT3
0540 CHAIN STAT4
0550 CHAIN STAT5
0560 CHAIN STAT6
0570 CHAIN STAT7
0580 CHAIN STAT8
0590 CHAIN STAT9
0600 CHAIN STAT10

THE GHOST
```

```
CHAINSTAT2

                    STAT2  -  HELP PROGRAM
STATPACK IS USED BY ENTERING DATA AT THE TERMINAL ROW
BY ROW USING STAT1 PROGRAM. YOU MUST STATE HOW MANY ITEMS THERE
ARE PER ROW AND PER COLUMN. THE NUMBER OF ITEMS WILL DEPEND ON
HOW MUCH MEMORY YOU HAVE. SAVED DATA CAN BE STORED ON DISC AND
USED OVER AND OVER BY DIFFERENT PROGRAMS.

      PRESS RETURN WHEN READY ?

YOU CAN DO VARIOUS CALCULATIONS AS FOLLOWS:

STATISTICS                      BASIC             3
AVERAGES        MEANS                             3
                                GEOMETRIC         3
                                HARMONIC          3
                                MEDIAN            3
                                MOVING            4
                                PROGRESSIVE       5
OTHER BASICS                    STD. DEVIATION    3
                COEFFICIENT OF VARIATION          3
                                HIGH-LOW          3
                                VARIANCE          3
                                RANGE             3
CHI-SQUARE                                        6
BAR CHARTS                                        7
SPARE PROGRAM                                     8
RELATIONSHIPS                                     9
REGRESSION                      COEFFICIENT OF    10
                                CORRELATION       10
                                LINEAR            10

NEW DATA ENTRY                                    1
INFORMATION                                       2


     ENTER PROGRAM NUMBER YOU REQUIRE ? 4



     ENTER PORT # ? 1
                    MOVING AVERAGE PROGRAM

     ENTER 'Y' FOR VISUAL OF DATA? Y

3 2 0 0 2 3 1 1 50 5 0 0 0 1 0 0 0 0 0
3 1 0 0 2 4 1 1 100 2 0 2 0 0 0 0 0 1 0
3 1 0 1 0 1 1 1 100 3 3 2 0 1 0 1 0 1 1
3 1 0 1 0 2 1 1 2 1 0 2 0 0 0 1 0 1 0
3 1 0 0 2 3 1 3 2 1 0 0 0 0 1 1 0 0 0
3 1 1 0 0 2 1 1 50 2 0 0 0 0 1 1 0 1 0

YOUR ARRAY IS 6 X 19. 114 ITEMS.
     RETURN IF READY?

                    MOVING AVERAGE

     COLUMN # STATISTICS REQUIRED FOR ? 6


     ENTER # OF UNITS ? 3
               1 TO 3          2.66666666
               2 TO 4          2.33333333
               3 TO 5          2
               4 TO 6          2.33333333



     ENTER 'Y' FOR MORE, 'N' FOR NONE ? N
                    STATPACK END

THE GHOST
#
```

which will follow are part 3, Basic Statistics, part 4, Moving Averages, part 5, Progressive Averages, part 6, Chi-Square, part 7, Bar-charts, part 8, a spare program for you to make up your own statistical package routine, part 9, Relationships between figures as used in questionnaires and part 10, Regressions and correlations.

Part 4, Moving Averages are used to smooth out data peaks and troughs in a series of data and the sample run shown gives an idea of what the results look like. Column 6 shows a variation between 1 and 4 but when smoothed to take account of seasonal variations you can see that the variations are between 2 and 2.6666. This indicates that there was a low period in the data between items 3 & 5 and the low of period 3 itself should be taken into consideration with other periods and not by itself.

This is the simplest part of the package and you will see that the program starts with an opening set of lines between 5 and 1280. Subroutines to get the data from the disc are at 9000 onwards. Lines 9600 to 9690 get the data and allow you to view it if required. The actual statistical part of the program is between lines 3000 and 3130.

In part 1 line 1170 should be altered to read as follows:–

1170 A3 = VAL(A£): IF A3 = 0 THEN 1240

It only affects CHAINing to section 10, which was written afterwards.

(Articles for the other parts will follow in good time).

```
LIST      STAT4          MICRO AID

0005 REM STAT4 MOVING AVERAGE PROGRAM FOR STATPACK
0020 INPUT "      ENTER PORT # ",Z9
0080 LINE= 132: GOSUB9380
0090 GOSUB 9380
0100 ? TAB(22);"MOVING AVERAGE PROGRAM":?
1010 GOSUB 9600
1050 ? "YOUR ARRAY IS ";A;"X ";B;:?CHR$(8);:?". ";A*B;"ITEMS."
1110 INPUT "      RETURN IF READY",A$
1120 GOSUB 3000
1200 GOSUB 9360
1210 IF LEFT$(A$,1)="N"THEN1240
1220 IF LEFT$(A$,1)="Y"THEN1260
1230 IF LEFT$(A$,1)<>"N"THEN1200
1240 ? TAB(20);"STATPACK END": END
1260 INPUT "      ENTER Y FOR MOVING ",A$
1270 IF A$="Y"THEN1120
1280 CHAIN STAT1
3000 ? #(Z9):?#(Z9),TAB(20);"MOVING AVERAGE"
3010 GOSUB 9400
3020 A4=0:A5=0
3030 IF B2<1THENRETURN
3040 ? :INPUT"      ENTER # OF UNITS ",B3
3050 IF B3>A-1THEN3040
3060 J1=0: FORI=1TOA-B3+1:A4=0
3070 J1=J1+1
3080 FOR J=J1TO(B3+J1-1)
3090 A4=A4+C(J,B2)
3100 NEXT J:A5=A5+1
3120 ? #(Z9)," ",I;"TO ";I+B3-1,A4/B3
3130 NEXT I: GOSUB9300: RETURN
9000 REM SUB PROGRAMS
9300 FOR K=1TO5:?#(Z9):NEXTK:RETURN
9360 INPUT "      ENTER `Y´ FOR MORE, `N´ FOR NONE ",A$:RETURN
9380 ? CHR$(25);:?CHR$(25);:?CHR$(22);:?CHR$(12);:RETURN
9400 IF B=1THENB2=1:GOTO9430
9410 ? :INPUT"      COLUMN # STATISTICS REQUIRED FOR ",B2
9420 IF B2>B THEN?"TOO HIGH";:GOTO9410
9430 ? #(Z9):RETURN
9600 OPEN #10, STATFL1 FOR INPUT: FIELD#10,F=6
9610 OPEN #20,STATFL2 FOR INPUT:FIELD#20,A=6,B=6
9640 SET #10=1:SET#20=1:GET#20
9650 DIM C(A,B),B3(A)
9660 FOR I=1TOA:FORJ=1TOB:GET#10:C(I,J)=F:NEXTJ:NEXTI
9683 INPUT "      ENTER `Y´ FOR VISUAL OF DATA",X$:IFX$<>"Y"THEN9690
9685 ? :FORI=1TOA:FORJ=1TOB:?C(I,J);:NEXTJ:?:NEXTI:?
9690 CLOSE #10:CLOSE#20:RETURN

THE GHOST
#
```

PCW OPEN PAGE

The Amateur Computer Club View

Mike Lord

DEF (REALITY)

AMATEUR One who practices esp. art or sport as a pastime (Little Oxford Dictionary)

"But don't you find the word 'amateur' in the ACC's name a disadvantage?" (Enthusiastic salesman)

Q: "But what do you *do* with your computer?"

A: "But what do you do with your golf ball?"

"S-100 Bus compatible? These busses are really all the same" (Enthusiastic salesman)

APPEND (LIST)

Readers living in the *Exeter* area may be interested to hear of a local group which has been formed by Trevor Brownen of Crystal Electronics, 40 Magdalene Road, Torquay.

Mr. Braga, of 3 Troutbeck Crescent, Bramcote, Beeston, Notts has offered to host a *Nottingham* group — write or 'phone him on 0602 256622.

The Thames Valley Group is still growing rapidly, and regular meetings are now being held on the first Thursday of each month at 7.30 pm at The Griffin, 10 Church Road, Caversham, *Reading*. For further details contact Dave Howland (0628) 36976 or Tim Hill (0734) 27812.

Similarly, the recently formed *Bristol* Group are now meeting at 7.30 pm on the third Wednesday of each month at Bristol Polytechnic Computer Centre, Coldharbour Lane, Frenchay. Rex Godby is organising events and would like to hear from anyone willing to give a talk or demonstrate their system. His address is 16 Williamson Road, Bristol BS7 9BH, tel: Bristol 46981.

Also in *Bristol,* a group is being set up at Brunel Technical College, Ashely Down Road, by Mr. S. Rabone of the Electrical and Electronic Engineering Dept. It will have access to the department's equipment. Mr. Rabone's telephone number (at the college) is 41241 ext 10 or ext 75.

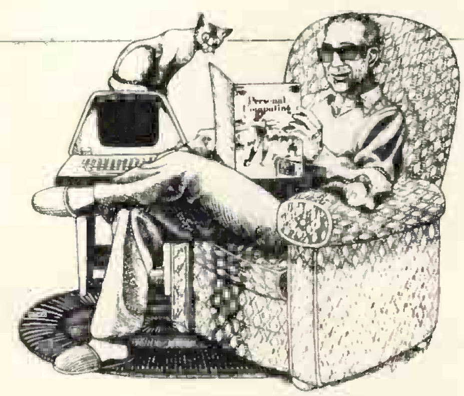
Peter Wearce intends to start a club in the *Brighton* area. Contact him at 30 Warren Way, Telscombe Cliff, Newhaven, East Sussex BN9 7D5.

Finally, a *Medway* Group has been formed by Tony Aylward of 194 Balmoral Road, Gillingham, Kent (Medway 56830).

And, of course, those wishing to know more about the National Amateur Computer Club are invited to send an s.a.e. to 7 Dordells, Basildon, Essex.

FREE PCW SERVICE FOR AMATEURS
BUY
SELL
EXCHANGE
EQUIPMENT, IDEAS, SOFTWARE
Maximum insertion: 100 words

SEND YOUR NOTICE TO
PCW OPEN PAGE
62a WESTBOURNE GROVE
LONDON, W2, ENGLAND.



3-D Noughts and Crosses

BILL DAVY

Department of Electrical Engineering
The University of Aston

The game of three dimensional noughts and crosses is a logical extension of the familiar two dimensional form. The aim is for the player to form a straight line of noughts before his opponent (in this case a computer) can form a straight line of crosses. The 3D game is played on a 4 x 4 x 4 board; there are therefore 64 points and 76 possible winning lines, the 8 central points and 8 corner points have 7 lines running through them and the other 48 points have 4 lines running through them.

Computer memory is a linear array, that is to say a cell in memory has a unique address and the two adjacent cells have addresses one greater and one less than that address. We have to represent the board of play in memory, and preferably in a representation which preserves the structure we are concerned with, in this case straight lines. We can name a point on the board by specifying its three coordinates, namely its level (L), its column (C) and its row (R) where each coordinate can take on a value of 0, 1, 2 or 3.

States

In this program the state of the point whose coordinates are l c and r is represented by the contents of the cell whose address is given by:— p=16*l + 4*c+r. This representation relates each of the 64 points on the board to a unique number in the range 0 to 63, and for convenience I shall call this number the index of the point. Table 1 shows the index calculated for each of four points. It was pointed out above that whatever representation of the board was chosen, it must provide a simple way of getting at points on a straight line. The points specified in table 1 do in fact lie on a straight line, and the indices increase by a constant amount between each point. By resorting to some elementary Cartesian geometry it can be shown that this property is always true, namely that points equidistant along a straight line have equal differences in their coordinates and this leads to equal differences in their indices.

This property of the indices for points on a straight line provides us with an easy way to store all the information relating to the 76 straight lines. Two additional arrays (A and B, see line 110) are set up to hold the starting indices and index differences (lines 910-970) for all the lines. By pre-computing and storing these values the execution time of the program is significantly reduced at the expense of a moderate increase in storage, a very common tradeoff in programming. The data for these two arrays was obtained from a program written in Coral 66 running on a TMS9900 microprocessor.

The Algorithm

Having obtained a convenient representation of the board and a simple method of storing all the necessary data about the lines, we need to consider an algorithm (set of explicit rules) by which the computer can determine, if not the best, at least a good place to go. This problem can be reduced to two parts: first work out the value of going on each unoccupied point, and then go on the point with the highest value.

Level	Row	Column	Index
0	3	3	16*0+4*3+3=15
1	2	3	16*1+4*2+3=27
2	1	3	16*2+4*1+3=39
3	0	3	16*3+4*0+3=51

TABLE 1 Indices for 4 Points on a Straight Line

Although this simplifies the problem, it does not provide a solution since not even the term 'value' is defined. An obvious aim for the computer is to maximise the number of lines with just crosses on, and to minimise the number of lines with just noughts on. This aim can be refined as: it is better to put a cross on a line which already has a cross on it (or better two crosses, and better still three) and it is better to put a cross on a line which already has a nought on it (or better two noughts, and better still three). This can be expressed simply as follows:—

"For each line do:—

Count the number of crosses (call it C) and the number of noughts (call it H)
If neither C nor H are zero then go to the next line
Obtain a value of V from table 2
For each unoccupied point on the current line, add V to its value"

This is the set of rules that the program uses to calculate the value of going on each unoccupied point. The values in table 2 are quite arbitrary, but they are greater for partially occupied lines, and greater for lines partially occupied by noughts thereby making the program slightly aggressive. The values of V for H=0,1,2,3 are stored (being read from line 980) in an array NØ, and the values of V for C=0,1,2,3 are stored (being read from line 990) in an array N1.

Another feature of the algorithm above is that table 2 is only used when there is a possible winning line. By counting the number of accesses (in T) to table 2 (actually arrays NØ and N1) it is possible to see if the game is drawn. In addition, line 660 prints out T as an indicator of how the game is progressing. It is interesting (though it is not done in the accompanying program) to print out M at line 725, as this is the computer's estimate of the value of its go.

C	V	H	V
0	1	0	0
1	4	1	3
2	9	2	8
3	16	3	15

TABLE 2 Line Weightings

The program

The program itself is very simple, largely because I intend to rewrite it in assembly language for the aforementioned microprocessor. For this reason I have avoided multiplication and division (other than by powers of 2) except for line 530 where $C^*H \neq 0$ is a simple way (in Basic) of testing to see if C and H are both nonzero. Another point to notice is that if your version of Basic expects (or allows) arrays to have a zero subscript then the '+1' that occurs in each subscript can be omitted. Finally, although throughout this article I have started counting at zero, this is not very natural for people. Since the program is to compete against humans it will subtract one from the coordinates read in (by subtracting 21 from the human go index in line 330).

I leave you with three questions. Can you obtain better (integer) values for the arrays NØ and N1 (i.e. for table 2)? Can you extend the principle and program to the game of 4D noughts and crosses? Can you find a better algorithm?

```
0100 RESTORE
0110 DIM A(75+1),B(75+1)
0120 DIM G(63+1),V(63+1)
0130 DIM NO(3+1),N1(3+1)
0140 READ A
0150 READ B
0160 READ NO
0170 READ N1
0180 FOR I=0 TO 63
0190 G(I+1)=0
0200 NEXT I
0210 FOR I=0 TO 63
0220 V(I+1)=0
0230 IF G(I+1)=0 THEN 250
0240 V(I+1)=-1
0250 NEXT I
0260 GOSUB 770
0270 PRINT"LEVEL:";
0280 INPUT L
0290 PRINT"COLUMN:";
0300 INPUT C
0310 PRINT"ROW:";
0320 INPUT R
0330 P=16*R+4*L+C-21
0340 IF V(P+1)=-1 THEN 270
0350 G(P+1)=-1
0360 V(P+1)=-1
0370 GOSUB 770
0380 N=-1
0390 T=0
0400 FOR L=0 TO 75
0410 S=A(L+1)
0420 V1=B(L+1)
0430 C=0
0440 H=0
0450 P=S-V1
0460 FOR I=0 TO 3
0470 P=P+V1
0480 IF G(P+1)<>1 THEN 510
0490 C=C+1
0500 GOTO 530
0510 IF G(P+1)<>-1 THEN 530
0520 H=H+1
0530 NEXT I
0540 IF H*C<>0 THEN 650
0550 T=T+1
0560 IF H=4 THEN 1030
0570 P=S-V1
0580 FOR I=0 TO 3
0590 P=P+V1
0600 IF V(P+1)=-1 THEN 640
0610 IF C<>3 THEN 630
0620 N=P
0630 V(P+1)=V(P+1)+NO(H+1)+N1(C+1)
0640 NEXT I
0650 NEXT L
0660 IF N<>-1 THEN 1050
0670 IF T=0 THEN 1100
0680 PRINT T;" POSSIBLE WINNING LINES BEFORE MY GO"
0690 M=-1
0700 FOR I=0 TO 63
0710 IF M>=V(I+1)THEN 740
0720 M=V(I+1)
0730 P=I
0740 NEXT 1
0750 G(P+1)=1
0760 GOTO 210
0770 FOR R=1 TO 4
0780 FOR L=1 TO 4
0790 FOR C=1 TO 4
0800 P=16*R+4*L+C-21
0810 IF G(P+1)<>-1 THEN 830
0820 PRINT"0";
0830 IF G(P+1)<>1 THEN 850
0840 PRINT"X";
0850 IF G(P+1)<>0 THEN 870
0860 PRINT".";
0870 NEXT C
0880 PRINT" ";
0890 NEXT L
0900 PRINT
0910 NEXT R
0920 RETURN
0930 DATA 0,0,0,1,2,3,3,4,8,12,0,0,1,2,3,3,0,0,1,2,3,3,4,4,5,6,7,7,8,8
0940 DATA 9,10,11,11,12,12,13,14,15,15,12,12,13,14,15,15,16,16,16,17,18
0950 DATA 19,19,20,24,28,32,32,32,33,34,35,35,36,40,44,48,48,48,49,50
0960 DATA 51,51,52,56,60
0970 DATA 1,4,5,4,4,3,4,1,1,1,16,17,16,16,15,16,20,21,20,20,19,20,16,17
0980 DATA 16,16,15,16,16,17,16,16,15,16,12,13,12,12,11,12,16,17,16,16
0990 DATA 15,16,1,4,5,4,4,3,4,1,1,1,1,4,5,4,4,3,4,1,1,1,1,4,5,4,4,3,4
1000 DATA 1,1,1
1010 DATA 0,3,8,15
1020 DATA 1,4,9,16
1030 PRINT"YOU WIN"
1040 GOTO 180
1050 PRINT"I WIN"
1060 G(N+1)=1
1070 GOSUB 770
1080 PRINT
1090 GOTO 180
1100 PRINT"GAME DRAWN"
1110 GOTO 180
1120 END
```

```
0100 REM INITIALISE DATA
0110 REM ALLOCATE START AND INDEX ARRAYS
0120 REM ALLOCATE GAME AND VALUE BOARDS
0130 REM ALLOCATE HUMAN AND COMPUTER LINE ARRAYS
0140 REM INITIALISE START INDEX ARRAY
0150 REM INITIALISE INCREMENT INDEX ARRAY
0160 REM INITIALISE HUMAN LINE ARRAY
0170 REM INITIALISE COMPUTER LINE ARRAY
0180 REM CLEAR GAME BOARD
0190 REM ,,
0200 REM ,,
0210 REM MARK OCCUPIED POINTS ON VALUE BOARD
0220 REM ,,
0230 REM ,,
0240 REM ,,
0250 REM ,,
0260 REM DISPLAY GAME BOARD
0270 REM PROMPT FOR LEVEL OF HUMAN'S GO
0280 REM INPUT SAME
0290 REM PROMPT FOR COLUMN OF HUMAN'S GO
0300 REM INPUT SAME
0310 REM PROMPT FOR ROW OF HUMAN'S GO
0320 REM INPUT SAME
0330 REM CALCULATE INDEX OF HUMAN'S GO
0340 REM ASK AGAIN IF ALREADY OCCUPIED
0350 REM PUT HUMAN'S GO ON GAME BOARD
0360 REM AND MARK VALUE BOARD
0370 REM DISPLAY GAME BOARD
0380 REM POSSIBLE INDEX OF COMPUTER'S FORCING WINNING GO
0390 REM INITIALISE POSSIBLE WINNING LINE COUNT
0400 REM FOR ALL LINES DO :-
0410 REM START INDEX FOR CURRENT LINE TO S
0420 REM INDEX INCREMENT FOR CURRENT LINE TO V1
0430 REM INITIALISE COMPUTER OCCUPIED POINT  COUNTER
0440 REM INITIALISE HUMAN OCCUPIED POINT COUNTER
0450 REM "INITIAL" POINT IS BEFORE LINE
0460 REM FOR EACH POINT ON LINE DO :-
0470 REM CALCULATE INDEX OF NEXT POINT
0480 REM IF NOT OCCUPIED BY COMPUTER,TRY HUMAN
0490 REM INCREMENT COMPUTER OCCUPIED POINT COUNTER
0500 REM GO TO NEXT POINT
0510 REM IF NOT OCCUPIED BY HUMAN, GO TO NEXT POINT
0520 REM INCREMENT HUMAN OCCUPIED POINT COUNTER
0530 REM NEXT POINT
0540 REM IF LINE HAS NOUGHTS AND CROSSES ON IT, IT HAS NO VALUE
0550 REM INCREMENT POSSIBLE WINNING LINE COUNTER
0560 REM IF 4 HUMAN POINTS ON LINE THEN HUMAN WINS
0570 REM "INITIAL" POINT IS BEFORE LINE
0580 REM FOR ALL POINTS ON LINE DO :-
0590 REM CALCULATE INDEX OF NEXT POINT
0600 REM NO VALUE IF OCCUPIED
0610 REM FORCED WIN ON 3 COMPUTER OCCUPIED POINTS
0620 REM REMEMBER FORCED WIN LOCATION IN CASE GAME NOT LOST
0630 REM INCREASE VALUE OF POINT
0640 REM NEXT POINT
0650 REM NEXT LINE
0660 REM IF FORCED WIN INDEX SAVED, THEN GO TO IT
0670 REM IF NO POSSIBLE WINNING LINES, THEN GAME DRAWN
0680 REM
0690 REM SET MAXIMUM VALUE BELOW MINIMUM POSSIBLE
0700 REM FOR ALL POINTS DO :-
0710 REM IF CURRENT VALUE BELOW MAXIMUM SO FAR, GO TO NEXT POINT
0720 REM SAVE NEW MAXIMUM VALUE
0730 REM SAVE INDEX OF MAXIMUM VALUE
0740 REM NEXT POINT
0750 REM PUT COMPUTER'S GO AT BEST POINT
0760 REM HUMAN'S TURN NEXT
0770 REM SUBROUTINE TO DISPLAY GAME BOARD
0930 REM LIST OF START INDEX FOR ALL LINES
0940 REM ,,
0950 REM ,,
0960 REM ,,
0970 REM LIST OF CORRESPONDING INDEX INCREMENTS
0980 REM ,,
0990 REM ,,
1000 REM ,,
1010 REM LIST OF VALUES FOR HUMAN OCCUPIED LINE
1020 REM LIST OF VALUES FOR COMPUTER OCCUPIED LINE
1030 REM CONGRATULATIONS !
1040 REM START GAME AGAIN
1050 REM BAD LUCK
1060 REM PUT COMPUTER'S FORCED WIN ON BOARD
1070 REM DISPLAY BOARD
1080 REM THROW A BLANK LINE
1090 REM START GAME AGAIN
1100 REM GAME DRAWN IF NO POSSIBLE WINNING LINES
1110 REM START GAME AGAIN
1130 END
```

The Poor Man's Printer

Phil Cornes

THE TYPE 7

Well, we've finally made it. The beginning of the age of the Personal Computer is upon us, and yet I can't help but wonder how many more people would take an active interest if only —

1. the initial cost of a computing system could be reduced, and

2. if the input and output could be made more like English and less like high level maths.

Unfortunately, the very act of finding a solution to point 2 by providing an input/output (I/O) peripheral capable of handling English, usually makes the problem, outlined in point 1, many times worse.

This is the position that I was faced with some short time ago, and the rest of this article is devoted to a brief explanation of the solution on which I finally settled.

The Type 7 Teleprinter

At first sight the type 7 teleprinter may appear as though it has too many disadvantages to be used as an I/O device. The biggest of these is that it doesn't use ASCII code, followed closely by the fact that it only uses a 5 bit code to define its character set (32 possible binary combinations) and therefore doesn't appear to have the ability to define the full alpha-numeric (letters, digits and special characters) range of characters. These disadvantages, as we shall see, can be easily overcome, and when you also take into account the fact that an ex-equipment type 7 can be purchased for around £30, including carriage, from several well-advertised distributors, you will realise that the type 7 is a good solution to an otherwise expensive problem.

I am probably very fortunate, in that I already come into vague contact with teleprinters in the course of my work, and therefore finding some "gen" on them was relatively simple. The first piece of much needed information is the character set and the International 5 unit teleprinter code needed to transmit it (for this, see Table 1). The second piece of information is to find out how an adequate character set is built up around only 32 unique codes. This is achieved as follows.

CODE NO	LETTERS	MSB 5	4	3	2	LSB 1	FIGURES
1	A	0	0	0	1	1	—
2	B	1	1	0	0	1	?
3	C	0	1	1	1	0	:
4	D	0	1	0	0	1	WHO ARE YOU
5	E	0	0	0	0	1	3
6	F	0	1	1	0	1	% (optional)
7	G	1	1	0	1	0	@ (optional)
8	H	1	0	1	0	0	£ (optional)
9	I	0	0	1	1	0	8
10	J	0	1	0	1	1	BELL
11	K	0	1	1	1	1	(
12	L	1	0	0	1	0	)
13	M	1	1	1	0	0	.
14	N	0	1	1	0	0	,
15	O	1	1	0	0	0	9
16	P	1	0	1	1	0	0
17	Q	0	1	0	1	0	1
18	R	0	1	0	1	0	4
19	S	0	0	1	0	1	'
20	T	1	0	0	0	0	5
21	U	0	0	1	1	1	7
22	V	1	1	1	1	0	=
23	W	1	0	0	1	1	2
24	X	1	1	1	0	1	/
25	Y	1	0	1	0	1	6
26	Z	1	0	0	0	1	+
27	carriage return	0	1	0	0	0	carriage return
28	line feed	0	0	0	1	0	line feed
29	letter shift	1	1	1	1	1	letter shift
30	figure shift	1	1	0	1	1	figure shift
31	space	0	0	1	0	0	space
32	all spaces	0	0	0	0	0	all spaces

TABLE 1

Each of the first 26 codes (as shown in Table 1) is assigned 2 distinct characters so that, for example, the letter E and the figure 3 both share the same coding of 00001. The teleprinter is able to distinguish between them because it contains a flip flop (mechanical) which responds to the receipt of either a letter shift, or figure shift (characters 29 and 30) preceding the characters to be printed. Once a letter shift has been received, all the characters subsequently printed will be letters until a figure shift is identified and vice versa. You will also notice from Table 1 that codes 27 through 32 provide the same function, *regardless* of the state of the flip flop. This gives you a total of 55 unique characters excluding figure shift, letter shift and all space (NOP).

Transmission

The next thing to consider is the teleprinter's method and speed of transmission.

The type 7 teleprinter transmits and receives data at a rate of 50 Bauds (bauds = bits/sec) which, when you consider that there are at least 8 bits in a character, gives a maximum transmission speed of just over 6 characters per second, each bit having a 20mS time slot. The reason that there are 8 bits to a character is

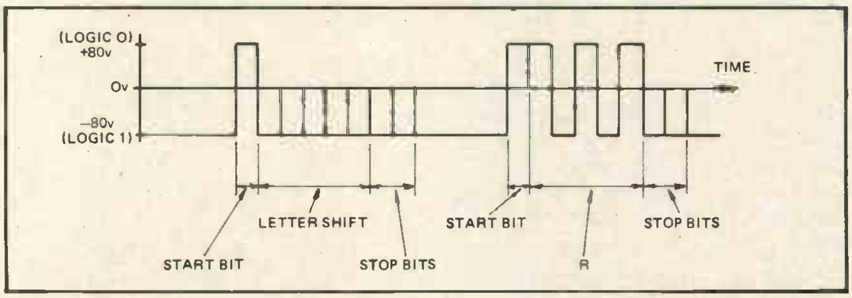


FIG. 1.

due to the fact that the 5 bit code is always preceded by a start bit (logic 0) and followed by at least 2 stop bits (logic 1). This can be more easily understood if we show an example diagramatically (letter shift followed by the letter R) (Fig. 1). You should not be put off by the fact that the teleprinter gives out +ve and –ve 80v on its transmit line, as this can easily be converted to TTL levels (or any other levels) by the addition of the simple single relay circuit of Fig. 2.

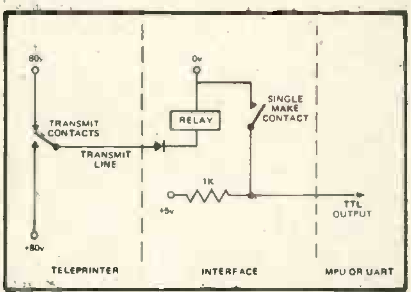


FIG. 2.

A similar circuit is required to perform the same function in reverse to enable the MPU to output data to the type 7 on its receive line. You should have noticed from the above that the type 7 has separate transmit (X) and receive (R) lines, so that in effect the keyboard connected to the transmit line, and the receive line connected to the printer, form two totally separate units, which can be used independently if required. This enables you to use the teleprinter in one of several modes.

1. LOCAL In this mode, the X and R lines are connected together being at the same time disconnected from the Interface and MPU. This can be useful when preparing paper tape or mag tape listings of new programs for loading later if, for example, the MPU is presently engaged in some other activity.

2. SIMPLEX In this mode, the X and R lines are again connected together, but this time connection is also made with the Interface and MPU. Connection in this way allows you to see a hard copy version of anything that you input to the MPU, as the transmitted information is routed straight back to the printer.

3. DUPLEX In this mode, the X and R lines are both connected to the MPU, but separately. This method of connection is useful if you wish to be able to check that the MPU has received the data from a keyboard or tape entry correctly. This is done by writing a routine to enable the MPU to directly output on the type 7 R line, any data that it receives on the X line, thus giving the operator a visual check that the character was received by the MPU correctly. This facility is called Echo.

Right! We now know a fair amount about the teleprinter. Let's go and see about running it into and from an MPU. There are one or two points which might cause a little bit of difficulty in this respect. The first of these is the 32 combination teleprinter code, because the MPU, in outputting data to the type 7, has to be able to generate the relevant letter and figure shifts. Consider the following example in BASIC –

10 PRINT "X2="

For those of you who know no BASIC, this statement is telling the computer to output to the I/O peripheral (type 7 here) whatever is contained within the inverted commas, so that in this case the following would appear from the type 7 printer –

X2=

This is all very nice, but in the case of the type 7 there would also have to be one letter shift and one figure shift in there as well, put in, in such a way that the MPU 'knew' when to make the change.

Easy, I hear you say. Enter them into the inverted commas just as any other character, as follows –

"L–S X F–S 2="

This would certainly work, but just consider how many bytes of memory it would require just in letter and figure shifts for the full statement given above –

F–S 10 L–S PRINT F–S "L–S X F–S 2="

There are five figure and letter shifts in what would normally have been a twelve byte statement. Quite a substantial increase in memory requirement! So I set about trying to find a better way, and ended up with the following.

My approach

When the MPU reads a character from the type 7, it is always preceded by a start bit and always followed by at least 2 stop bits. This just nicely fills an 8 bit register somewhere within the MPU, so that when it writes a character to the type 7, it just regurgitates the code as it was input. If we now examine closely the 8 bits to be output, we find that one of them is redundant. That is, bit 8, the second stop bit. We need to output the start bit to tell the 7 that it is about to receive a character (being asynchronous in this respect) and to give the mechanism time to start up. We also need the 5 bit code to define the character, but the two stop bits are merely provided to allow the printer mechanism time to restore at the end of the cycle, and to make sure that the X or R lines do not show a start bit when the mechanism has restored (or the printer will start a new cycle). Therefore, we only have to output the first stop bit (which will be latched onto the output line anyway) and make sure that it lasts at least 40mS before the MPU tries to write another character.

If we now set up a status bit in the MPU, the state of which depends upon the letter and figure shifts, and write this status bit into bit 8 of every character as it is input, then

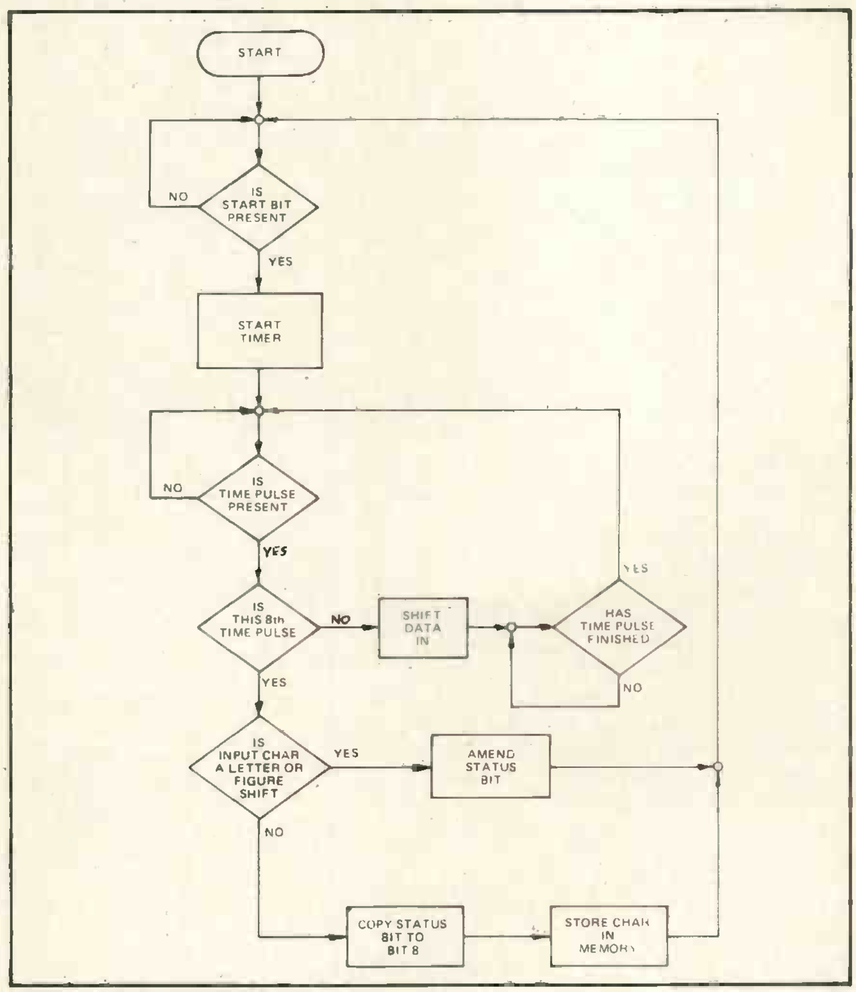


FIG. 3. – INPUT ROUTINE

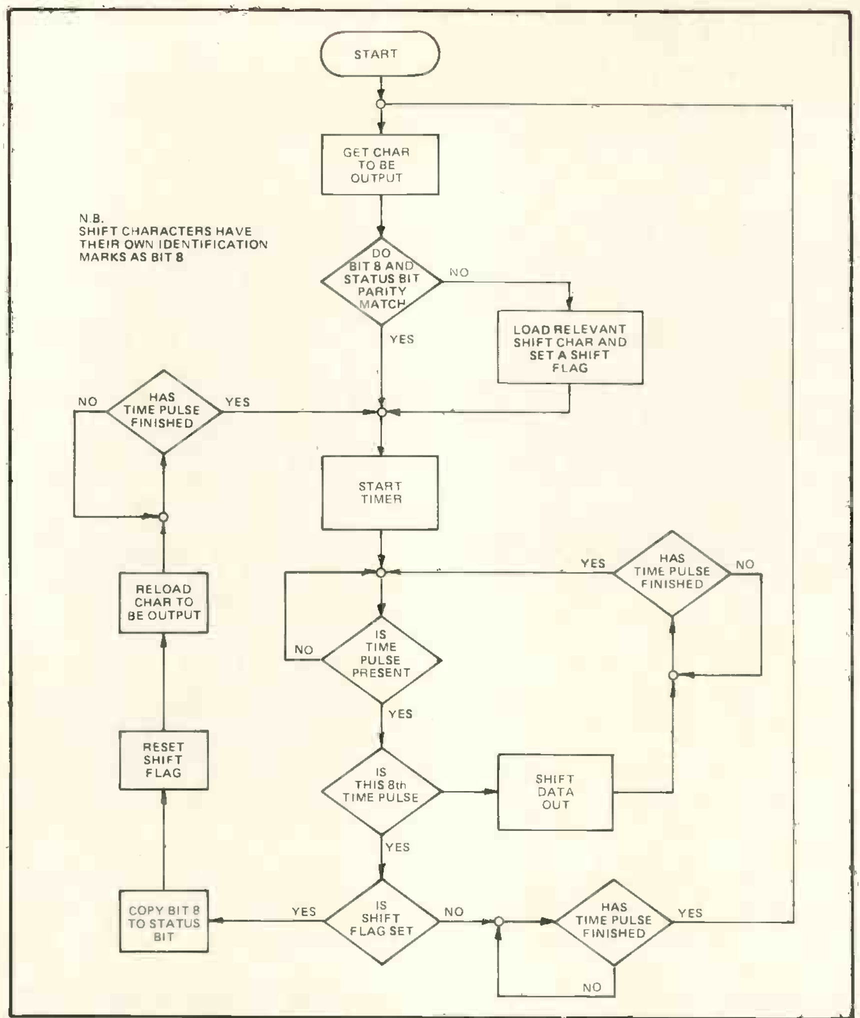


FIG. 4. — OUTPUT ROUTINE

each byte of data will contain its own letter and figure shift identification mark. In this way, the letter and figure shift characters need no other form of storage.

All that is required now, when we output data, is to make a comparison between the 8th bit of each character before it is printed, and the status bit (which is made equal to the 8th bit of the previous character printed) to see if they are the same. When they are not, then we have to output the relevant shift character before printing continues.

Possible flow charts for routines to perform these functions appear as figs. 3 and 4. A 20mS synchronising pulse generator is required to work in conjunction with these routines, so that to receive a character the MPU scans the X line until it sees a start bit. When this is detected, the MPU passes a start pulse to an external timer which provides a series of 8 scan pulses, the leading edges of which are 20mS apart. The first 7 of these tell the MPU when to shift serial data from the X line into itself, and in the period of the 8th pulse the MPU calculates and loads the 'shift' bit into the 8th bit place.

When the MPU has data to output, this is loaded into a serial register or UART. The 8th bit is compared with the status bit for parity, a shift character is loaded and output, if necessary, then the character to be printed is output. This is achieved by starting the external timer and shifting data out of the register as the leading edges of the first 7 scan pulses are received. The 8th pulse gives the MPU time to copy the 8th bit to the status bit. After the trailing edge of the 8th timing pulse, the MPU "knows" that the type 7 has restored and can now output another character, if there is one.

You would think that after wading through that lot, this would be just about the end of it. Unfortunately, this is not the case. There is one other very important item that we have yet to look at.

The Maths

When writing an interpreter (say for BASIC) there is no more problem in getting an MPU to recognise the words LET or PRINT etc in teleprinter code than there is in recognising the same words in ASCII code, because they are still just 8 bit binary combinations.

One problem which does arise, however, is with the maths. An MPU usually does all its calculations in strict Binary or in BCD. Using ASCII code this presents no problems, because the 8 bit codes for the digits 0 through 9 already contain the binary (or BCD) combinations of these digits; so that all we need to do is to extract the correct 4 bits from the code.

With the teleprinter code (see table 1) this is not the case. The digits appear to be randomly distributed within the 32 possible combinations in such a way that the codes bear no binary resemblance to the digits that they represent.

Solving it

There are several methods of implementing a solution to this problem.

You could buy a teleprinter (Baudot) code to ASCII code converter (expensive) or you could 'instruct' the MPU such that when it received certain combinations it would translate them into binary and reverse the procedure to output answers (wasteful of memory) or you could do what I am doing — *modify the teleprinter.*

You will know, if you have ever seen the guts of a teleprinter, that to alter the code sent out when any particular key is depressed would be practically impossible to perform, let alone describe. There is, however, an easier way — swap the labels round on the keyboard!

This can easily be done such that none of the lower case characters (letters) are altered, so you still end up with a QWERTY keyboard. The

CODE NUMBER	FIGURES	CODE NUMBER	FIGURES
1	—	17	7
2	9	18	?
3	:	19	'
4	WHO ARE YOU	20	0
5	£ (optional)	21	+
6	% (optional)	22	=
7	@ (optional)	23	3
8	4	24	/
9	)	25	5
10	BELL	26	1
11	(	27	CARRIAGE RETURN
12	2	28	LINE FEED
13	.	29	LETTERS
14	,	30	FIGURES
15	&	31	SPACE
16	6	32	ALL SPACE

TABLE 2

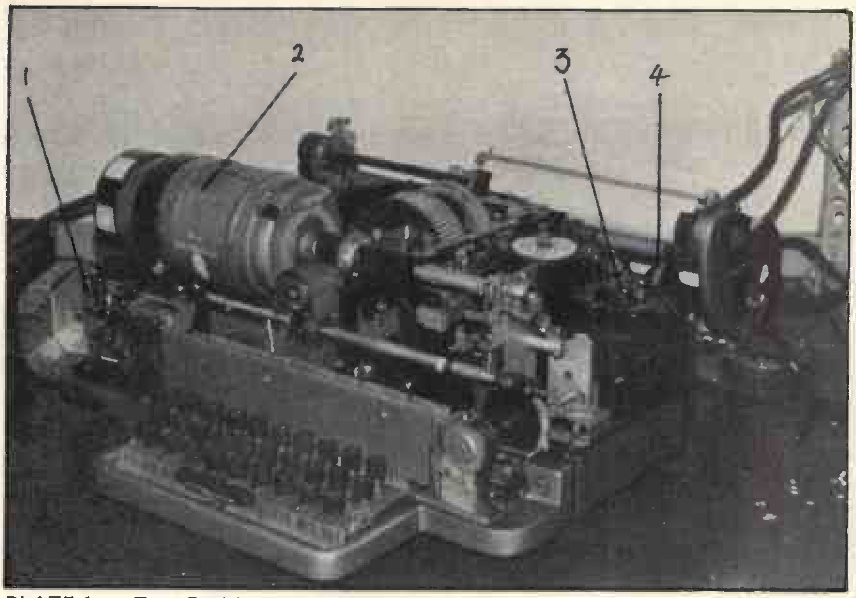


PLATE 1 — Type 7 with case removed

KEY — 1. TRANSMIT CONTACTS
2, DRIVE MOTOR
3. RECEIVE MAGNET
4. CARRIAGE RELEASE LEVER

following changes represent the simplest method of getting a binary code.

0 to 5, 5 to 6, 6 to 0
1 to +, + to 7, 7 to 1
2 to),) to 8, 8 to 9, 9 to ?,
? to 4, 4 to £, £ to 3, 3 to 2.

Table 2 gives a list of the revised figure shift characters as they appear following the modification. The binary bits appear as bits 1 (LSB) through 4 in the modified code so that all that has to be done now is to extract these 4 bits in the same manner as for the ASCII code. Now the only problem to be solved is that of getting the correct digit printed when the MPU outputs answers. This can be done in a similar manner to the above, by swapping the character heads round on the printing barrel in the same order that the keyboard labels have just been swapped.

This modification requires the removal of the printing barrel so that the heads can be manipulated. After removing the case of the teleprinter depress the carriage return key and rotate the motor by hand (clockwise as viewed from the left) so that the teleprinter performs this operation. This puts the printing barrel in the correct position for removal. Now locate and depress the carriage release lever (right hand side under the carriage mechanism). This allows you to hinge the complete carriage mechanism back until it hits a stop.

At this point, the complete mechanism should be lifted vertically away from the rest of the teleprinter and laid gently to one side. If you now look at the teleprinter from the rear you will see the barrel holding the print heads and in front of this you will see a support bracket in which the barrel rotates. The bracket is held onto the printer by one cheese headed screw which should be unscrewed. This will allow you to (carefully!) remove the bracket. The printing barrel can now be pulled straight out towards you. Looking at the front (the side you were looking at when you pulled it out) of the barrel you will see that the print heads are held in place by a circular plate which is in turn held in place by 3 small springs. At this point you can either remove the 3 springs altogether to allow you to remove the plate, or if you are careful you need only pull the plate out against the tension of the springs just enough to allow you to remove and relocate the necessary print heads. When the modification has been made, the teleprinter can be reassembled in the reverse order to that given for its dismantling.

PLATE 2 — Removal of the Printing Barrel

KEY — 1. PRINTING BARREL
2. RETAINING BRACKET
3. REMOVED CARRIAGE

As you can see from all of the above, anyone with £30 to spend and a few hours of time can produce an I/O peripheral whose facilities could cost up to several hundreds of pounds, using other equipment. All I can say now is if this article helps only a dozen people to realize the joys of the Personal Computer World, then it will have been worth the effort.

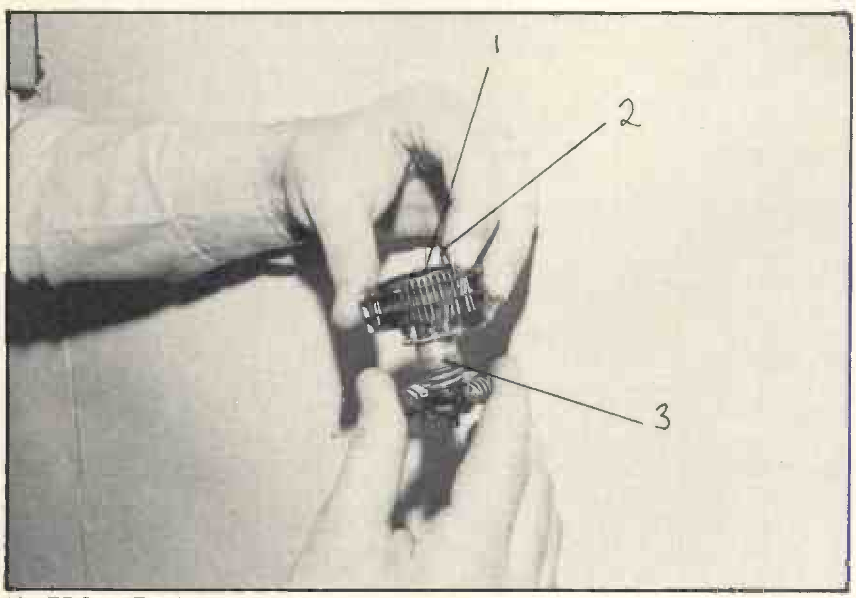


PLATE 3 — The Printing Barrel

KEY — 1. CIRCULAR RETAINING PLATE
2. TYPE HEAD BEING REMOVED
3. SHAFT

THE NEWBEAR COMPUTING STORE

7 Bone Lane, Newbury.

77-68 The Best Supported Hobbyist 6800 System in the U.K.

Available as Bearbags or as Individual Components.

Each Bearbag contains all the Components necessary to build a vital part of a micro-computer system. Each part is backed with the support necessary for such a complex project.

BEARBAG 1	77–68 CPU PCB and Components	£45.00
BEARBAG 2	77–68 LED's and Switches	£14.95
BEARBAG 3	77–68 Power Supply	£17.95
BEARBAG 4	77–68 5u Rack and Backplane	£27.70
BEARBAG 5	77–68 4K Ram PCB and Components	£75.00
BEARBAG 6	77–68 Mon 1 PCB and Components	£50.70
BEARBAG 12	77–68 VDU PCB and Components	£69.50
BEARBAG 13	77–68 Mon 2 PCB and Components	£t.b.a.
BEARBAG 16	77–68 EPROM Board and Components	£t.b.a.
BEARBAG 17	77–68 Interface Board (PIO) and Components	£t.b.a.

6800, Z80 16K Dynamics

DOWN IN PRICE

SYM·1

(Formerly VIM-1)

The new 6502 Micro from Synertek. Fully assembled and tested.

Ex. Stock. @ £199.00

+8% VAT, carriage £1.00

Floppy Disc Drives

5¼"	SHUGART SA400 DISC DRIVE	£225.00
	Carriage by Securicor 'C' Rate	£4.50
8"	DRI SINGLE SIDED DOUBLE DENSITY	£325.00
	Carriage by Securicor 'C' Rate	£5.00
8"	DRI DOUBLE SIDED DOUBLE DENSITY	£365.00
	Carriage by Securicor 'C' Rate	£5.00

NASCOM·1

The Z80-Based Microcomputer Kit.

Ex. Stock. @ £197.50

+8% VAT, carriage £1.00

3 Amp Power Supply £24.50

Send for our New Winter Catalogue

OTHER BEARBAGS AVAILABLE

BEARBAG 7	4K Ram PCB and Components (Exorcisor Compatible)	£71.50
BEARBAG 8	8K Ram PCB and Components (Exorcisor Compatible)*	£16.00
BEARBAG 9	Petitevid VDU Kit. V24/RS232C Interface. 64 Characters by 16 lines. Requires +5 –12 power supply, ASCII encoded. Keyboard, Video monitor of UHF Modulator (Bearbag 11) and a domestic T.V.	£85.00
BEARBAG 10	Kansas City Cassette Interface	£18.95
BEARBAG 11	UHF Modulator 12½% VAT	£ 4.50
BEARBAG 14	2708 Prom Programmer Kit (6800)	£35.00
BEARBAG 15	Promverter. Mikbug to 2708	£ 8.50
BEARBAG 18	Cottis Blandford Cassette Interface	£17.25
BEARBAG 19	TRS80 Level II 4K to 16K Byte Conversion	£115.00
BEARBAG 20	TRS80 Level I 4K to 16K Byte Conversion	£115.00
BEARBAG 21	Lower Case Kit for Petitevid.	£10.50

Postage and Packing 50p

*Postage and Packing £1.00

Add 8% VAT to all prices unless otherwise stated.

BEARBAGS ARE AVAILABLE FROM:–

NEWBEAR COMPUTING STORE
Bone Lane, NEWBURY. (Mail Order & Callers) 0635 49223

NEWBEAR COMPUTING STORE
2A Gateley Road, Cheadle, (Callers Only)

MICRODIGITAL
25 Brunswick Street, LIVERPOOL. Tel: 051-236 070

CRAYWORTH LTD.,
346 London Road, Blackwater, CAMBERLEY, Surrey.
Tel. CAMBERLEY 340044

CRYSTAL ELECTRONICS
40 Magdalenen Road, TORQUAY. Tel: 0803 22699

SINTROM MICROSHOP
14 Arkwright Road, READING, Berks. Tel: 84322

THE BYTE SHOP
426/428 Cranbrook Road, Gants Hill, ILFORD, Essex.
Tel: 01-554 2177

Send for our New Book List

Books

77-68 Design Manual	£ 7.50	.50p
Introduction to Microcomputers.		
Volume 0: The Beginners Book	£ 5.95	.50p
Volume 1: Basic Concepts	£ 5.95	.50p
Volume 2: Some Real Products June 77 rev.	£11.95	1.00
8080A/8085 Assembly Language Programming	£ 6.95	.50p
6800 Assembly Language Programming	£ 6.95	.50p
Some Common Basic Programmes	£ 5.95	.50p
6800 Programming for Logic Design	£ 5.95	.50p
8080 Programming for Logic Design	£ 5.95	.50p
Payroll with Cost Accounting in Basic	£ 9.95	1.00
Understanding Microcomputers and Small Computer Systems	£ 7.56	.50p
Scelbi 6800 Software Gourmet Guide and Cookbook	£ 7.95	.50p
Scelbi 8080 Software Gourmet Guide and Cookbook	£ 7.95	.50p
8080 Standard Assembler	£15.95	.75p
The Scelbi Byte Primer	£ 9.95	1.00
The 8080 Programmers Pocket Guide	£ 2.35	.30p
Microprocessors C201	£ 8.00	.50p
Microprocessor Interfacing Techniques	£ 8.00	.50p
Guide to SC/MP Programming	£ 3.75	.50p
SC/MP Assembly Language Programming Manual	£ 3.75	.50p
SC/MP Microprocessor Applications Handbook	£ 3.75	.50p
Micro – The 6502 Journal	£ 1.70	.30p
Sym Reference Manual	£ 7.50	.75p
Sym Programming Manual	£ 7.50	.75p
First Book of Kim	£ 7.50	.75p
Kim 1 User Manual	£ 7.50	.75p
6500 Hardware Manual	£ 7.50	.75p
6500 Programming Manual	£ 7.50	.75p
101 Basic Computer Games	£ 5.50	.50p
Games, Tricks and Puzzles for a Hand Calculator	£ 2.49	.50p
Games with Pocket Calculators	£ 1.75	.30p
Star Ship Simultation	£ 5.10	.50p
Beginning Basic	£ 2.95	.50p
Introduction to Basic	£ 1.95	.50p
Some Common Basic Programmes	£ 5.95	.50p
Payroll with Cost Accounting In Basic	£ 9.95	1.00
Instand Freeze and Dried Programming in Basic	£ 4.95	.75p
My Computer Likes me When I speak in Basic	£ 1.65	.30p
Computer Programmes that Work	£ 2.40	.75p
What you do After You Hit Return	£ 7.00	.75p
Illustrated Basic	£ 1.90	.50p
Beginning Basic	£ 4.95	.50p
Learning Basic Fast	£ 6.30	.50p
8080A Microprocessor Family Quick Reference	£ 1.50	.50p

ALL MAIL ORDERS TO NEWBURY PLEASE

24 Hour Turnaround on Orders.
Barclaycard and Access welcome. No Minimum. Overseas orders issued with a Pro-forma invoice. Minimum Official Order £10.00. Send for catalogue to Newbear Computing Store, Bone Lane, Newbury, Berks. Callers welcome Mon. – Sat. 9.00 – 5.30, *But please phone us first* on 0635 49223. New Office – 2A Gateley Road, Cheadle, Cheshire. Tel: 061 491 2290.

A High Speed Cassette Interface

Bob Cottis and Mike Blandford

Mass Storage

One of the main advantages of a computer is the ability to easily change the function of the device by simply loading a new program. In the simplest case this is done by typing in a new set of instructions by hand. This soon gets pretty boring, even for a short BASIC program, but imagine loading the BASIC interpreter by hand each time you wanted to use it! Clearly some form of mass storage is an urgent requirement for any computing system. In the past mini-computer systems have tended to use paper tape as a (relatively) cheap storage device, but paper-tape equipment tends to be fairly noisy and expensive. Consequently the most popular mass storage medium for personal computing is magnetic tape cassettes. With care low cost audio recorders can give reliable recording and playback.

Having decided that cassette recorders are a good thing, how should the data be recorded?

The first thing to do is to convert the parallel data byte (eight bits on eight lines) to a serial form (eight bits one after the other on a single line). This can either be done synchronously, whereby a continuous stream of data is transmitted, or asynchronously, in which timing information precedes and follows each eight bits of data. For various reasons, particularly its tolerance of speed and other occasional errors, an asynchronous format is generally used with audio cassette recorders. An exception to this is the Tarbell interface, which uses synchronous recording. In general this is reported to operate reliably, but should it 'drop' a bit and get out of synchronization the rest of the record is garbled.

Waveform

The waveform for the asynchronous transmission of a byte of data is shown in fig. 1. In the idle condition (no data being transmitted) the signal is high. It goes low for one bit period to signify that data is on its way. This is followed by up to eight bits of data, least significant bit first. This may be followed by a parity bit which can be used to check the received data for errors. Finally the signal is held high for one or two bit periods to signify the end of the data. This also ensures that the receiver will see the high to low transition required to indicate a new start bit. This is all pretty complicated, but fortunately the semiconductor manufacturers have done all the work for us, and devices known as UARTs (Universal Asynchronous Receiver Transmitter) are readily available. A common pin-out is shown in fig. 2. The various control inputs allow the UART to be set up to give five to eight bits of data, odd, even or no parity and one or two stop bits. Status outputs indicate what the device is up to. The timing of the transmitter and receiver sections is controlled by two clock inputs, which must be at 16 times the data rate. The UART receiver samples the incoming data at what it expects to be the middle of each data bit. If the clock is too fast or slow the sampling point will move towards the start or finish of the data bit. Depending on the word length the clock must be accurate to about 5 to 9% for correct reception.

Recording

Now that the data has been turned into a tidy string of high and low logic levels, it must be converted to a

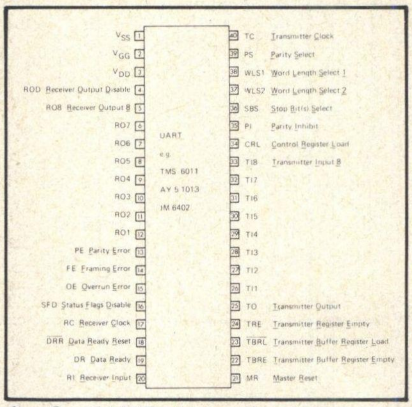


fig. 2. UART pin configuration

suitable form for recording. Direct recording of the UART output is not satisfactory owing to the wide variation of the DC level of the signal. The next simplest approach is to switch an audio tone on for a one and off for a zero. Unfortunately the automatic level controls fitted to most cheap recorders get a bit unhappy when fed with this sort of signal. Also the need to detect the tone requires a fairly long burst, and hence a rather low data rate. The next step is to use two tones, one for a one and the other for a zero. This is the basic principle of CUTS, a format sponsored by BYTE magazine to provide a standard for reliable, interchangeable cassette recording. This uses tones at 2400 and 1200 Hz. A *one* consists of eight cycles at 2400 Hz, while a *zero* consists of four cycles of 1200 Hz. These frequencies were chosen to give a standard 300 baud data rate, and to allow for easy regeneration of the 16 times clock (4800 Hz) for the UART. In theory the high redundancy (using several flux transitions to convey each bit of information) gives a highly reliable recording, but in practice few receiver circuits make full use of the redundancy.

CUTS is still slow, an 8K byte program taking about four minutes to load in binary format, or ten minutes in a hexadecimal format. In

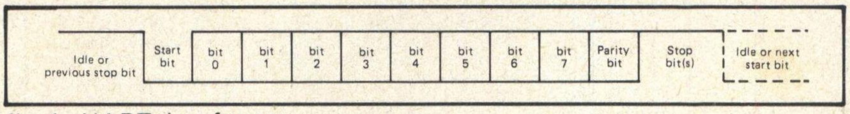


fig. 1. UART data format

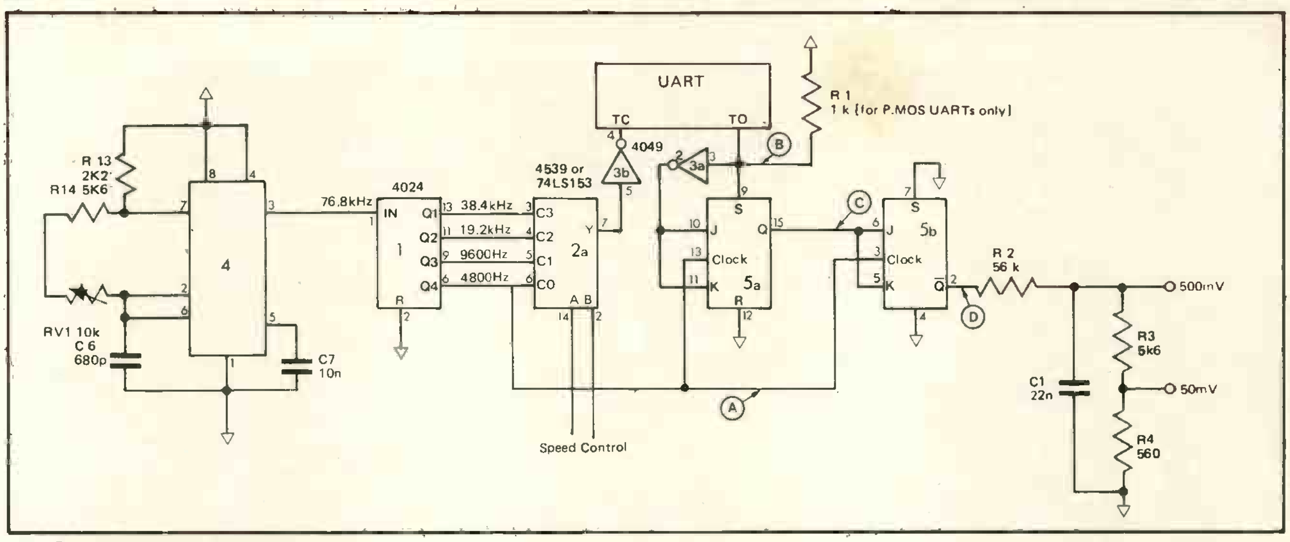


fig. 3. Transmitter Circuit.

order to speed up the recording the number of cycles of 1200 and 2400 Hz can be reduced by successive factors of two to obtain 600, 1200 and 2400 baud. At the maximum speed a one consists of 1 cycle of 2400 Hz, and a zero is half a cycle of 1200 Hz.

This interface can operate at all of these speeds from standard CUTS at 300 baud to 'high speed CUTS' at 2400 baud.

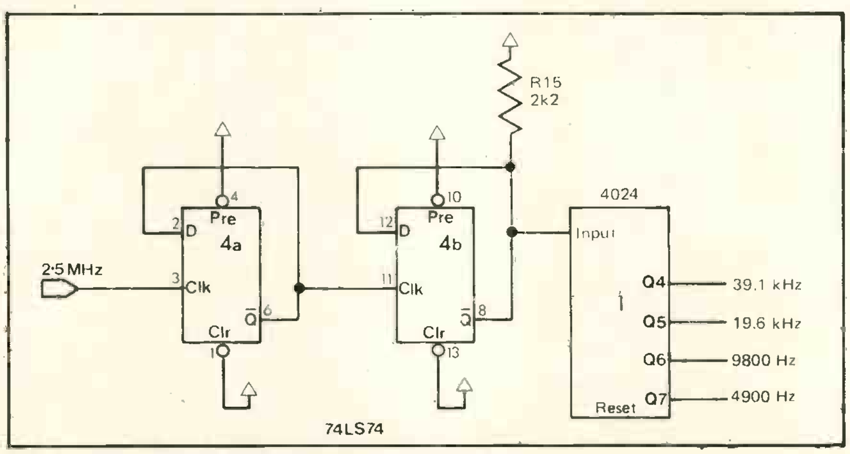


fig. 4. Deriving Master Oscillator from 2.5 MHz

The Transmitter

The transmitter circuitry is shown in fig. 3. The timing of the entire interface is controlled by a master oscillator. In fig. 3. this is shown as a 555 operating at 76.8 kHz, but it can be derived from any stable source, a crystal being ideal. In particular a Z80 system running at 2.5 MHz can give a master clock within 1% of the correct frequency. A suitable circuit is shown in fig. 4.

The 16 times clock for the UART is selected by the multiplexer, IC2a, which is controlled by the two speed control inputs. For standard CUTS the frequency of this clock is 4800 Hz, giving a transmitter output at 300 baud. This output controls whether the two J-K flip-flops, IC5, divide the 4800 Hz clock input by two or four, giving the required CUTS format. This is put through a simple low pass filter to remove the fast edges of the square wave. For higher speed recording the multiplexer selects a faster clock for the UART, which shortens the duration of each bit. The transmitter waveforms at 2400 baud are shown in fig. 5.

The Receiver

Putting the data onto tape is fairly easy, the hard part is getting it back again. The first thing to do is to get rid of as much extraneous noise as possible (particularly mains hum), and then convert the signal to a nice clean square wave. This is achieved by IC3c and IC10a (fig. 6) in conjunction with the associated circuitry. C2 and RV1 form a high pass filter which attenuates low frequency signals such as mains hum. The signal is then amplified by a CMOS inverter IC3c, used in the analog mode (use the unbuffered version if possible). The resultant signal is squared by the schmitt trigger, IC10a. The DC conditions can be adjusted with RV1 such that the input to the schmitt trigger is in between the two switching levels in the absence of an input signal. The two exclusive-or gates (IC10b and c) then form a transition detector, which gives a negative-going output pulse each time the input changes state. These pulses are inverted and used to trigger a 'digital monostable' (IC9a). Providing this monostable is triggered (i.e. the counter is reset) before 12 clock pulses have been counted, the input to the D-type flip-flops (IC7) remains high, and the inverted output from the second flip-flop (IC7b) remains high. This will be the case whenever a 2400 Hz input signal is being received. With a 1200 Hz input signal the counter will reach 12 about 3/4 of the way through the half cycle, taking the output of the NAND gate (IC6d) low. The next pulse from the transition detector takes the output from IC7a and IC3d low. Two more 2400 Hz pulses from the transition detector are required to send the inverted output of IC7b high again.

This serves to extend the duration of the low output to the correct time.

Feeding the UART receiver clock

Having extracted the data, we must now obtain a signal at 16 times the receiver bit rate to feed to the UART receiver clock. In a perfect world this could be the same as the transmitter clock. This would probably work if the same recorder were always used for both recording and playback, but in general low cost cassette recorders tend not to have very accurate tape speeds. Consequently this approach can give difficulties in playing back tapes recorded on other machines. Fortunately, the signal from the recorder contains sufficient informa-

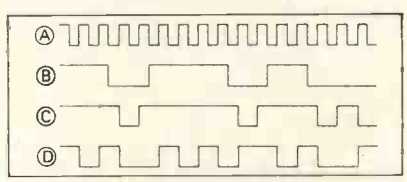


fig. 5. Transmitter Waveforms

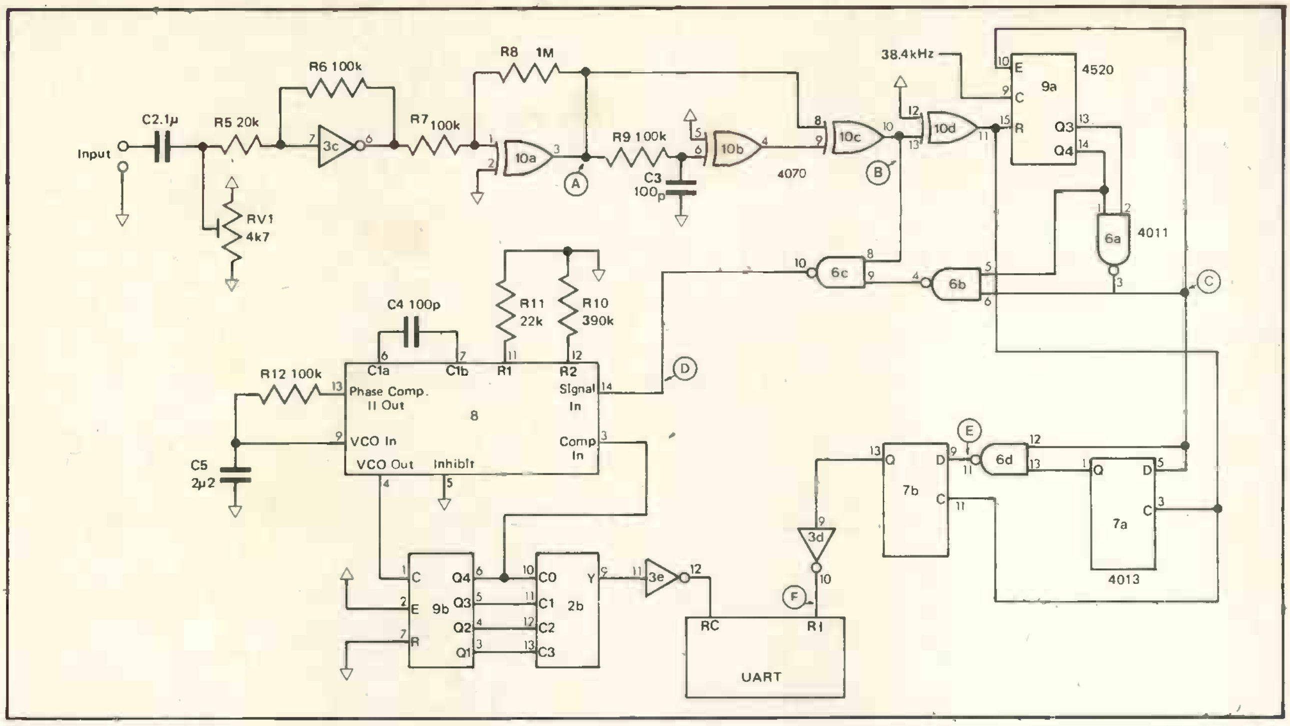


fig. 6. Receiver Circuit

ation to reconstruct the clock frequency. When a 2400 Hz signal is being received the transition detector will give an output at 4800 Hz. With a 1200 Hz signal the transition detector will give an output of only 2400 Hz, so an extra pulse must be found from somewhere. Such a pulse is available from the counter (IC 9a) which is acting as a digital monostable. Nominally this counter will reach eight exactly half way through the half cycle. This is all we need for CUTS, but at the higher speeds we need a higher frequency.

Once again, the semi-conductor makers come to our aid with a device known as a phase locked loop (PLL). For our purposes this can be considered as a device which will adjust its output frequency to keep two input frequencies the same. Obviously one of the input frequencies must be derived from the output to keep the loop stable, but a divide by N counter can be fitted between the input and the output, whereupon the output frequency will be N times the signal input frequency. Thus the phase locked loop can, in effect, multiply an input frequency by any whole number. In this circuit the phase locked loop is designed to operate at 76.8 kHz and the required signals of 38.4, 19.2, 9.6 and 4.8 kHz can be obtained from the feedback divider, IC9b. The correct frequency is selected by the multiplexer, IC2b, in the same way as for the transmitter, and fed to the 16-times clock of the UART. The inverting buffer provides a higher drive capability for use with P-MOS UARTs. The waveforms at various points in the circuit are shown in fig. 7.

Connecting the Interface to Your Computer

This interface can be used with a ready made board which already contains a UART. All that is needed is to connect the transmitter and receiver clocks and the serial in and out from the UART as shown in figs. 3 and 6. The two speed control bits can be connected to an existing output port if software selectable speeds are required, or a 2 pole, 4 way switch can be used for manual speed control, as shown in fig. 8.

If the UART is also being added to the system, several options are available, depending on how much control you want the computer to have over the UART. In the most basic system the computer must be able to read and write data for the UART, and also read the UART status, particularly DR (Data Ready). The various control inputs such as the number of stop bits and word length must be hard-wired to logic 1 or 0, and the UART allowed to see them by wiring CRL (Control Register Load) to logic 1.

At times, however, it may be useful to be able to change these options, and this can be done by addressing the control register as an output port. If one of the control inputs, such as WLS2, is hard-wired, this leaves four spare lines on the control port, and these can be latched by a 74(LS)175 or 74(LS)75 to be used to select the recording speed, and to switch two reed relays for recorder control. This allows two separate recorders to be used, one for recording and one for playback. With a bit of thought on the software, this will permit large blocks of text, such as assembler source listings, to be handled without needing massive amounts of memory. An example of this approach can be seen in Patrick Sutton's article in PCW Vol. 1, No. 2.

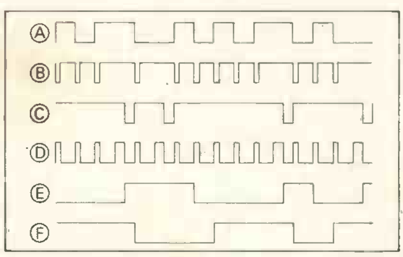


fig. 7. Receiver Waveforms

One way of connecting the UART is shown in fig. 9. This uses the E78 bus conventions, but only minor changes should be needed to adapt the circuit to other bus structures. The 74LS138 is actually decoding four read and write addresses, and a second UART can be fitted with no extra decoding. This could be used to provide a second cassette interface, or to give a V24 interface for a VDU, Teletype or similar device.

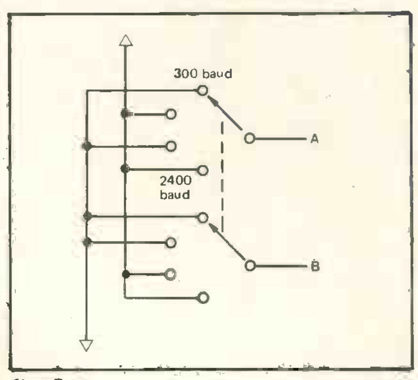


fig. 8.
Switch Selection of Operating Speed

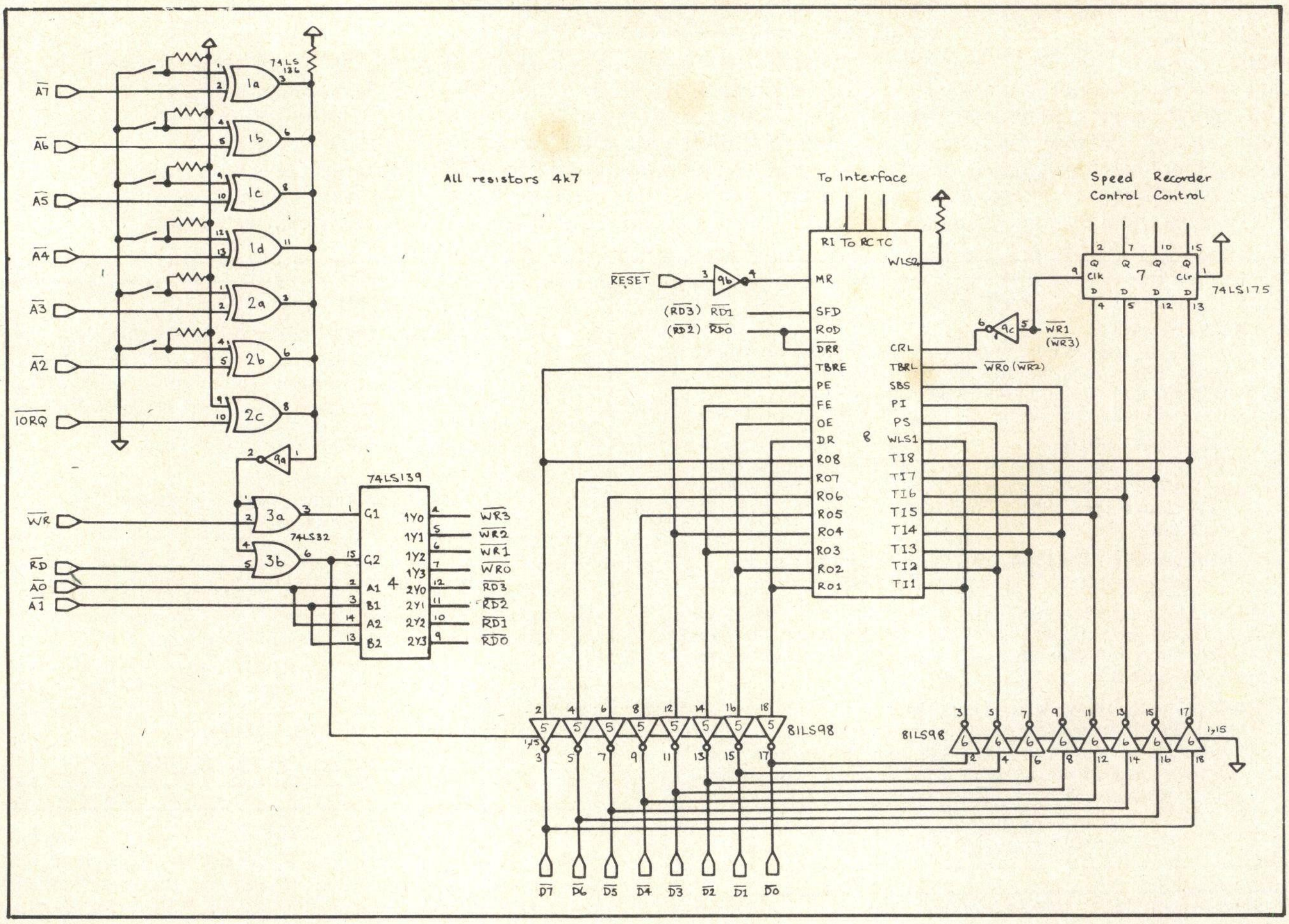


Fig. 9

Does It Work?

As far as we can tell this system is very reliable, even at 2400 baud. The few errors that have occurred have generally been associated with particular patches of tape, especially in the first foot or so of a cassette. This system does record at a very high density, and high quality tape should be used. It is inadvisable to use tapes longer than C60, and cassettes should be carefully stored away from dusty atmospheres etc. The cassette recorder heads should be kept clean.

Because it extracts its clock from the incoming data this circuit is very tolerant of speed variations. With the prototype the master oscillator frequency could be adjusted from 30% below to 13% above its nominal value. While a recording was being made the frequency was swept from one extreme to the other. The frequency was reset to its nominal value, and the tape was read with no errors.

Construction

Providing the normal precautions are taken to avoid discharging static electricity into the CMOS devices, there should be no problems in constructing the circuit.

The only adjustments required are to set the master oscillator frequency and to set the DC input conditions with RV1. The latter is just a matter of looking at the output of IC10a with a meter or logic probe, and "twiddling" RV1 to about half-way between the two points at which it changes state. The gain of the input should be adequate for most cassette recorders, if required it can be adjusted by changing R6. The gain is simply R6/R5 (e.g. the gain as shown is five). If a crystal oscillator is used there are no further adjustments required to the circuit, although the best tone and volume settings of the recorder should be found by trial and error. If an RC oscillator is used this must be adjusted to give an idle output frequency of 2400Hz. This should be done as accurately as possible (for the sake of friends with less speed tolerant CUTS circuits), though tuning slightly sharp of top D on a piano should give reasonable results.

The circuit, being CMOS, should not be sensitive to layout, and can be constructed on Veroboard or by wire-wrapping. For lazy constructors a small circuit board carrying all the circuit except for the UART can be obtained from **Newbear.** Detailed instructions on connecting this board to the Nascom microcomputer are also available. An E78 circuit board is also being prepared, and will include this interface, a second UART with a standard V24 interface, and space for wiring additional input and output ports. This again should be available from Newbear.

For The Experimenter

Just as any program can be improved, so this circuit is by no means the ultimate cassette interface. One intriguing possibility is that the output from the phase locked loop of the receiver could be used to drive the digital monostable. This would allow the receiver to track an even wider range of frequencies. Unfortunately it also gives a system with several stable states, and it doesn't work with the circuit as it stands.

Another possibility is that of running at even higher data rates by increasing the clock oscillator frequency and adjusting the phase locked loop. The prototype has operated for a short time at 5000 baud without errors, but you do this at your own risk!

Because of the inevitable variations between components, the resistor and capacitor values for the phase locked loop have been selected to give a fairly wide range over which the loop will lock. In this context it should be noted that all 4046s do not seem to be equal. The circuit was designed using a Motorola equation, and works using Motorola chips. Occasional chips from other makers may give trouble, which can be overcome by adjusting the range over which the loop will lock. First adjust the low frequency to 50kHz. Temporarily connect a 10 kΩ resistor from pin 9 of the PLL to ground. Measure the output frequency (if you don't have a frequency meter you can still measure the frequency by connecting the output of the feedback counter to a spare bit of an input port, and letting your computer do the work). The ideal resistor value is then the current value of R10 times the current frequency (in kHz) divided by 50. The upper frequency may be adjusted in a similar way to 110 kHz. Move the end of the 10 kΩ resistor from ground to +5V, measure the output frequency and adjust R11 by a factor of the measured frequency divided by 100 kHz. This calculation is not entirely accurate, so check both frequencies after making the adjustment.

If the output of the schmitt trigger is connected to a spare bit of an input port you can have a go at software decoding of other recording formats such as Tarbell.

Software Support Engineer

Personal Computers From £5500 Based Bedford

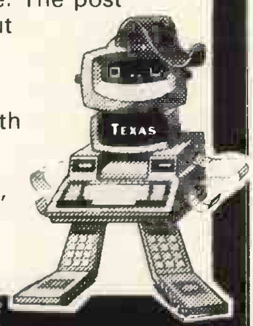
Texas Instruments needs no introduction as our programmable calculators and digital watches lead their fields.

Our Advanced Technology Consumer Products Team is committed to ambitious growth plans in personal computing and needs a software engineer to support our strategy in the U.K. This will involve applications development, product introductions and sales/dealer support, plus liaison with our central programming team.

Candidates should have experience of computer systems analysis, with small business systems for mini-computers. An interest in marketing would be an advantage. The post demands a highly committed and self-motivated person, not just a hobbyist – but someone who wants to build a career in personal computing.

Salary is negotiable from £5500 and could be substantially higher for the outstanding candidate. Attractive fringe benefits include generous assistance with relocation expenses, where appropriate.

Please write with full career details to Bob Denton, European Consumer Division, Texas Instruments Limited, Manton Centre, Manton Lane, Bedford, or telephone Ann Stevens, on Bedford (0234) 67466 ext. 3442 for an application form.

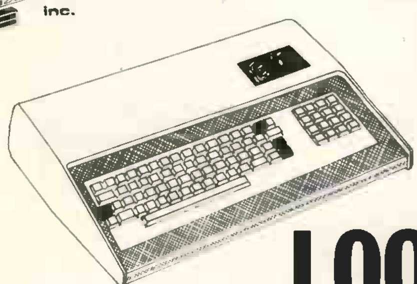


TEXAS INSTRUMENTS
LIMITED

Exidy inc.

Introducing the personal computer you've waited for.

The Exidy Sorcerer.



16K £760
32K £859
add 8% VAT

LOOK AT THESE FEATURES

* WORD PROCESSING, COBAL, FORTRAN etc
* PLUG IN ROM CARTRIDGES
* WORKS WITH NORMAL TV
* S100 EXPANSION UNIT
* CASSETTE INTERFACE
* Z80 CPU
* 32K RAM ON BOARD
* A REAL BUSINESS MACHINE

The Sorcerer Computer is a completely assembled and tested computer system ready to plug in and use. The standard configuration includes 63 key typewriter-style keyboard and 16 key numeric pad dual cassette I/O, with remote computer control at 100 and 1200 baud data rates, RS232 serial I/O for communication, parallel port for direct Centronics printer attachment, Z80 processor, 4K ROM operating system, 8K Micrsooft BASIC in separate plug-In Rom Pac™ cartridge, composite video of 64 chars × 30 lines, 128 upper/lower case ASCII character set and a 128 user-defined graphic symbols, up to 32K on-board RAM memory, operators manual, BASIC programming manual and cassette/video cables, connection for S100 bus expansion unit giving access to the spectrum of exciting and useful peripheral devices, such as Floppy disk drives, voice recognition/synthesis battery back-up board in case of power failure, additional memory boards, E-PROM cards give you the facility to program and re-program your own ROM memories etc. etc. This is the most useable and flexible system that's now available to the home and business user at such a low price.

please make cheques and postal orders payable to JADE
phone your order quoting ACCESS or VISA number
for technical information or advice phone 0736 66565

Supplied by – Factor One Computers

JADE
17 Market Place, Penzance, Cornwall.

PET Software

From our new catalogue of over 100 titles:

Microchess	£14.00
General Ledger	£20.00
Discounted Cash Flow	£ 8.00
VAT	£17.50
Payroll Suite	£25.00
6502 Disassembler	£15.00
Percentage Costing	£49.50
Bridge Challenger	£ 8.00
Income Tax 78/79	£ 8.00

Access/Barclaycard/Mastercharge accepted by telephone. If you are a PET owner and would like to receive regular copies of the catalogue, send an SAE to:

Petsoft

PO Box 9, Newbury, Berks. RG13 1PB
Tel. 0635-201131 01-352 1100 Telex 8951672

SWTPC 6800 MICROCOMPUTER AT A BARGAIN PRICE

Guaranteed in full working order and hardly used. 28k CPU with 3 serial interfaces: twin mini-floppy disc drives and controller: cassette interface: Soroc 1Q120 VDU: Teletype 43 printer: full documentation. Offers around £2,800. Individual units available at pro-rate prices.
Ring 01-398 7146

BUSINESS systems for as little as £3037. Tailor made software commissioned. SWTP products for personal and educational use also. Books and magazines available. Barclaycard accepted. VERWOOD SYSTEMS – Telephone Rugby 87629 for catalogue.

Support Your Local MICROPROCESSOR!

With our MM57109 Based Kit, add all the functions of a powerful scientific calculator to your Z80 or SC/MP based computer.

* Supplied complete with instructions and software to interface to NASCOM I Kit. No modifications to the NASCOM I are required.
* May require as little as 100(H) Bytes of ROM/RAM to operate.
* Your Microprocessor is left free to organise input and output of Data.
* Gives you direct control of computer programs – No need to work via high level languages.

COST: £39.75 + £1.25 p.p.

ONLY FROM:
CONTROL AND ELECTRONIC DEVELOPMENTS
BOX 7

6800 D2 MICROPROCESSOR DEVELOPMENT SYSTEM

Unused and in full working order. Consists of CPU board with user I/O interface, monitor and 384 bytes RAM: Keyboard/Display Board with cassette interface. Full documentation. Offers around £150.
Ring 01-874 9889

ANNOUNCING THE MICROS

JOIN THE MICRO REVOLUTION

£399 for a Z80 based microcomputer, built and tested

Designed for educational establishments, personal computing and small business users

* Includes 1K monitor Eprom, 47 key solid state keyboard, video, TV, cassette and teletypewriter interfaces,serial i/o, 2 parallel i/o ports, 2K bytes RAM, power supplies and instrument housing.
* Connect to domestic TV or video monitor to complete the system
* 48 x 16 character video matrix
* 47 key contactless ASC11 keyboard
* Hard copy on teletypewriter
* 2 TTL compatible parallel i/o ports
* RS232 serial i/o port
* Load and dump programmes on unmodified cassette recorder
* Up to 16K byte mixed RAM and Eprom in table top housing
* Expandable up to 64K bytes
* Security locked power switch
* British designed and built
* Available in kit form for £360
* Credit terms available

THE MICRONICS COMPANY
1, STATION ROAD TWICKENHAM MIDDLESEX
01-892 7044

PART OF THE MICRO REVOLUTION
Prices exclusive of VAT and carriage

THE SORCERER HAS ARRIVED

Introducing the personal computer you've waited for. The Exidy Sorcerer.

I didn't buy my personal computer until I found the one that had all the features I was looking for.
The Exidy Sorcerer does everything I wanted to do and a few things I never dreamed of.
It isn't magic. Exidy started with the best features of other computers, added some tricks of their own, and put it all together with more flexibility than ever before available. Presto! My reasons for waiting just disappeared.

I wanted pre-packed programs. Software on inexpensive cassette tapes for the Sorcerer is available from Exidy and many other software makers.

I wanted user programmability The Sorcerer's unique plug-in ROM PAC™ Cartridges contain programming languages such as Standard (Altair 8k*) BASIC, Assembler and Editor (so I can develop system software), operating systems such as DOS (so I can also use FORTRAN and COBOL) and applications packages such as Word Processor.

*Altair is a trademark of Pertec Computer Corp.

I wanted Graphics, and the Sorcerer is super. Its 256 character set — more than any other personal computer — includes 128 graphic symbols that I can define.

I wanted high resolution video. With 122,880 points in a 512 x 240 format, I get the most detailed illustrations.

I wanted to display more information. The Sorcerer displays 1920 characters in 30 lines of 64 characters — equal to a double-spaced typed page.

I wanted a full, professional keyboard. The Sorcerer's 79-key data processing keyboard provides designated graphics, the complete ASCII character set in upper and lower case, and a 16-key numeric pad.

I wanted memory. The 12k of ROM holds a Power-On Monitor and Standard BASIC; 32k of RAM is supplied on board.

I wanted expandability. Serial and parallel I/Os are built in, and the optional 6-slot S-100 expansion unit lets my system grow.

I wanted a computer that's easy enough for children to use. I just connect my Sorcerer to a video display and a cassette tape recorder, and if I have any questions the easy-to-understand Operation and BASIC Programming manuals have the answers.

I wanted to buy from an experienced Manufacturer. In five years Exidy has become the third largest producer of microprocessor-based video arcade games.

I wanted to spend less than £1,000. (This is where COMP. does a little magic). My Sorcerer cost me £950!.

Now, what are you waiting for?
Call COMP. on 01-441 2922 or write to

COMP COMPUTER COMPONENTS
14 Station Road,
New Barnet, Herts. EN5 1QW.

(Price shown ex. VAT)

DEALER ENQUIRIES INVITED, A LIMITED NUMBER OF DEALERSHIPS STILL AVAILABLE.